```
  SSSSSSSSSSSS  DDDDDDDDDDDD       AAAAAAAAA
  SSSSSSSSSSSS  DDDDDDDDDDDD       AAAAAAAAA
  SSSSSSSSSSSSS DDDDDDDDDDDD       AAAAAAAAA
SSS             DDD       DDD  AAA         AAA
SSS             DDD       DDD  AAA         AAA
SSS             DDD       DDD  AAA         AAA
SSS             DDD       DDD  AAA         AAA
SSS             DDD       DDD  AAA         AAA
SSS             DDD       DDD  AAA         AAA
   SSSSSSSSS    DDD       DDD  AAA         AAA
   SSSSSSSSS    DDD       DDD  AAA         AAA
   SSSSSSSSS    DDD       DDD  AAA         AAA
            SSS DDD       DDD  AAAAAAAAAAAAAAA
            SSS DDD       DDD  AAAAAAAAAAAAAAA
            SSS DDD       DDD  AAAAAAAAAAAAAAA
            SSS DDD       DDD  AAA         AAA
            SSS DDD       DDD  AAA         AAA
            SSS DDD       DDD  AAA         AAA
SSSSSSSSSSSS    DDDDDDDDDDDD   AAA         AAA
SSSSSSSSSSSSS   DDDDDDDDDDDD   AAA         AAA
SSSSSSSSSSSS    DDDDDDDDDDDD   AAA         AAA
```

```
_S

Sy
--
CT
CT
CT
CT
CU
CU
CU
DC
DC
DC
DC
DC
DC
DE
DE
DE
DE
DE
DE
DE
DE
DI
DI
DI
DI
DI
DI
DI
DI
DI
DI
DI
DI
DU
DU
DU
DU
DU
DU
DU
DU
DV
DV
DV
DY
DY
DY
DY
DY
ER
ES
```

```
**FILE**ID**DEVICE





DDDDDDDD    EEEEEEEEEE  VV      VV    IIIIII      CCCCCCCC  EEEEEEEEEE
DDDDDDDD    EEEEEEEEEE  VV      VV    IIIIII      CCCCCCCC  EEEEEEEEEE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EEEEEEEE    VV      VV      II      CC          EEEEEEEE
DD      DD  EEEEEEEE    VV      VV      II      CC          EEEEEEEE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE          VV      VV      II      CC          EE
DD      DD  EE            VV  VV        II      CC          EE              ....
DD      DD  EE            VV  VV        II      CC          EE              ....
DDDDDDDD    EEEEEEEEEE      VV        IIIIII      CCCCCCCC  EEEEEEEEEE      ....
DDDDDDDD    EEEEEEEEEE      VV        IIIIII      CCCCCCCC  EEEEEEEEEE      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

DE
VO

Table of contents

```
0000          .title  device  Display device data structures
0000          .sbttl  copyright notice
0000          .ident  'V04-000'
0000     ;
0000     ;****************************************************************************
0000     ;*                                                                          *
0000     ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000     ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000     ;*  ALL RIGHTS RESERVED.                                                    *
0000     ;*                                                                          *
0000     ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000     ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000     ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000     ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000     ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000     ;*  TRANSFERRED.                                                            *
0000     ;*                                                                          *
0000     ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000     ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000     ;*  CORPORATION.                                                            *
0000     ;*                                                                          *
0000     ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000     ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000     ;*                                                                          *
0000     ;*                                                                          *
0000     ;****************************************************************************
0000     ;
```

```
0000          .sbttl  Program description
0000    ;++
0000    ; Facility
0000    ;
0000    ;       System Dump Analyzer
0000    ;
0000    ; Abstract
0000    ;
0000    ;       This module contains routines to print device data
0000    ;       structures for the i/o subsystem.
0000    ;
0000    ; Environment
0000    ;
0000    ;       Native mode, User mode
0000    ;
0000    ; Author
0000    ;
0000    ;       Tim Halvorsen, July 1978
0000    ;
0000    ; Modified by
0000    ;
0000    ;       V03-011 EMB0110         Ellen M. Batbouta       24-Jul-1984
0000    ;               Fix a typo in the SHOW DEVICE display and update the
0000    ;               list of devices and device characteristics.
0000    ;
0000    ;       V03-010 EMB0105         Ellen M. Batbouta       07-Jun-1984
0000    ;               Add routines to display the contents of the class
0000    ;               driver data blocks (CDDB) when displaying an mscp
0000    ;               served device. Also for mscp served devices check
0000    ;               2 additional queues before drawing the conclusion
0000    ;               that the io request queue is empty.  Fix a minor
0000    ;               bug and include the node name in the display in
0000    ;               the routine, SHOW_SYSTEM_BLOCK.
0000    ;
0000    ;       V03-009 EMD0082         Ellen M. Dusseault      12-Apr-1984
0000    ;               Print the address of the cddb and the alternate cddb
0000    ;               (if the device is mscp served) when displaying the ucb tables
0000    ;               and action routines .  Also display the reasons to wait
0000    ;               count for mscp served devices.
0000    ;
0000    ;       V03-008 LMP0221         L. Mark Pilant,         30-Mar-1984  11:53
0000    ;               Change UCB$L_OWNUIC to ORB$L_OWNER and UCB$W_VPROT to
0000    ;               ORB$W_PROT.
0000    ;
0000    ;       V03-007 EMD0059         Ellen M. Dusseault      07-Mar-1984
0000    ;               Fill in local ucb with zeroes in routine, GET_UCB,
0000    ;               just in case next ucb fetched is shorter than the
0000    ;               previous one.
0000    ;
0000    ;       V03-006 WHM0002         Bill Matthews           16_Feb-1984
0000    ;               Change IDB$B_COMBO_VECTOR back to IDB$B_VECTOR.
0000    ;
0000    ;       V03-005 TMK0002         Todd M. Katz            29-Jan-1984
0000    ;               Add DT$_NI to the table BUS_TYPE.
0000    ;
0000    ;       V03-004 WHM0001         Bill Matthews           16-Jan-1984
0000    ;               Change IDB$B_VECTOR to IDB$B_COMBO_VECTOR.
```

```
0000  ;
0000  ;       V03-003 TMK0001         Todd M. Katz            19-Nov-1983
0000  ;               Change DT$_UNA11 to DT$_DEUNA in the table SCOM_TYPE and
0000  ;               add DT$_DECUA to the same table.
0000  ;
0000  ;       V03-002 ROW0237         Ralph O. Weber          10-OCT-1983
0000  ;               Enhance all displays for latest and greatest I/O database
0000  ;               information.  Add support for SHOW DEVICE/ADDR <expr>, where
0000  ;               expression is a UCB address.
0000  ;
0000  ;       V03-001 KTA3041         Kerbey T. Altmann       26-Apr-1983
0000  ;               Fix for cluster names.
0000  ;--
```

```
                  0000                 .sbttl  declarations
                  0000        ;
                  0000        ;        symbol defintions
                  0000        ;
                  0000                 $adpdef                     ; Adapter Control Block (ADP)
                  0000                 $aqbdef                     ; ACP queue header block (AQB)
                  0000                 $cddbdef                    ; Class Driver Data Block (CDDB)
                  0000                 $cdrpdef                    ; Class Driver Request Packet (CDRP)
                  0000                 $crbdef                     ; channel request block (CRB)
                  0000                 $dcdef                      ; device class/type definitions
                  0000                 $ddbdef                     ; device data block (DDB)
                  0000                 $ddtdef                     ; Driver dispatch table (DDT)
                  0000                 $devdef                     ; Device characteristics definitions
                  0000                 $dptdef                     ; Driver prologue table (DPT)
                  0000                 $dyndef                     ; Dynamic storage type definitions
                  0000                 $idbdef                     ; interrupt dispatch block (IDB)
                  0000                 $iodef                      ; I/O function codes
                  0000                 $irpdef                     ; I/O request package (IRP)
                  0000                 $mscpdef                    ; Mass Storage Control Protocol (MSCP)
                  0000                 $orbdef                     ; Object's Rights Block (ORB)
                  0000                 $pbdef                      ; path block (PB)
                  0000                 $pcbdef                     ; Process control block (PCB)
                  0000                 $sbdef                      ; System block (SB)
                  0000                 $tpadef                     ; TPARSE definitions
                  0000                 $ttyucbdef                  ; terminal UCB definitions
                  0000                 $ucbdef                     ; unit control block (UCB)
                  0000                 $vcbdef                     ; Volume control block (VCB)
                  0000                 $vecdef                     ; interrupt transfer vector (in IDB)
                  0000
                  0000        ;
                  0000        ; definition of requested device name storage fields
                  0000        ;  (using storage based at parsed_devnam)
                  0000        ;
                  0000                 $defini pdvnm
                  0000        $def     pdvnm_t_node    .blkb 16         ; node name
                  0010        $def     pdvnm_t_ddc     .blkb 16         ; device & controller
                  0020        $def     pdvnm_w_unit    .blkw 1          ; unit number
                  0022        $def     pdvnm_b_nodesz  .blkb 1          ; size of real node name
                  0023                                                  ;  (use by get_ddb)
        00000024  0023                 .blkb 1
        00000024  0024        pdvnm_k_length = .                        ; size of this structure
                  0024                 $defend pdvnm
                  0000
                  0000        ;
                  0000        ; definition of flags bits stored in r8 by display_device
                  0000        ;
                  0000                 _vield  flag,0,< -
                  0000                         <one_unit,,m>, - ; a specific unit was specified
                  0000                         <alt_path,,m>, - ; traversing the alternate DDB chain
                  0000                         <fnd_unit,,m>, - ; found at least one unit
                  0000                         >
```

```
                0000            .sbttl  storage definitions
                0000    ;
                0000    ;       storage definitions
                0000    ;
                0000
            00000000            .psect  sdadata,noexe,wrt
                0000
000000CC        0000    ucb_size = ucb$k_lcl_disk_length
                0000    .iif gt <ucb$l_2p_cddb+4-ucb_size>, ucb_size = ucb$l_2p_cddb+4
                0000
00000060        0000    sb:     .blkb   sb$k_length     ; System block (SB)
                0060    nodnam_2p:
00000071        0060            .blkb   sb$s_nodename+1
                0071
000000B5        0071    ddb:    .blkb   ddb$k_length    ; device data block (DDB)
000000F9        00B5    ddb_2p: .blkb   ddb$k_length    ; secondary device data block (DDB)
                00F9
000001C5        00F9    ucb:    .blkb   ucb_size        ; unit control block (UCB)
                01C5                                    ;  all the interesting stuff
                01C5
00000289        01C5    irp:    .blkb   irp$c_length    ; I/O request package (IRP)
                0289
00000331        0289    cdrp:   .blkb   cdrp$c_cd_len-cdrp$l_ioqfl      ; Class Driver Request Package (CDRP
000000A8        0331    cdrp_length=cdrp$c_cd_len-cdrp$l_ioqfl  ; Total length of cdrp including negative of
                0331
0000041D        0331    vcb:    .blkb   vcb$c_length    ; Volume control block (VCB)
                041D
00000439        041D    aqb:    .blkb   aqb$c_length    ; ACP queue header block (AQB)
                0439
00000471        0439    dpt:    .blkb   dpt$c_length    ; Driver prologue table (DPT)
                0471
000004E1        0471    cddb:   .blkb   cddb$k_length   ; Class driver data block (CDDB)
                04E1
00000551        04E1    cddb_2p:    .blkb  cddb$k_length ; Secondary CDDB
                0551
                0551    parsed_devnam:
00000575        0551            .blkb   pdvnm_k_length
                0575
                0575    flag_2nd_cddb:
    0000        0575            .word 0         ; flag to tell us if the address coming in is the
                0577                            ; primary or secondary cddb in routine, show_cddb
                0577    queue_notempty:
      00        0577            .byte 0         ; if 1 means item in an io queue to be displayed
                0578                            ; if 0 the queue is empty
                0578
            00000000            .psect  device,exe,nowrt,long
                0000
                0000            .default displacement,long
```

```
0000            .sbttl  read-only data definitions
0000
0000    ;
0000    ;       read-only data definitions
0000    ;
0000
0000    pb_status:
0000            table   pb$v_,<tim>
0010
0010    pb_state:
0010            table   pb$c_,<CLOSED,ST_SENT,ST_REC,OPEN>
0038
0038    pb_rstate:
0038            table   pb$c_,<UNINIT,DISAB,ENAB>
0058
0058    pb_rport_type:
0058            table   pb$c_,<CI780,HSC,KL10,CINT,NI,PS>
0090
0090    ddb_acpclass:
0090            table   ddb$k_,<PACK,CART,SLOW,TAPE>
00B8
00B8    unit_status:
00B8            table   ucb$v_,<tim,int,erlogip,cancel,online,power,timout,-
00B8                    inttype,bsy,mounting,deadmo,valid,unload,template,-
00B8                    mntverip,wrongvol,deleteucb,lcl_valid,supmvmsg,-
00B8                    mntverpnd>
0160
0160    device_char:
0160            table   dev$v_,<rec,ccl,trm,dir,sdi,sqd,spl,opr,rct,net,fod,-
0160                    dua,shr,gen,avl,mnt,mbx,dmt,elg,all,for,swl,idv,odv,-
0160                    rnd,rtm,rck,wck>
0248
0248    device_char_2:
0248            table   dev$v_,<clu,det,rtt,cdp,2p,mscp,ssm,srv,red,nnm>
02A0
02A0    device_class:
02A0            addr_table dc$_,<-
02A0                    <disk,disk_type>,-
02A0                    <tape,tape_type>,-
02A0                    <scom,scom_type>,-
02A0                    <card,card_type>,-
02A0                    <term,term_type>,-
02A0                    <lp,lp_type>,-
02A0                    <workstation,workstation_type>,-
02A0                    <realtime,realtime_type>,-
02A0                    <bus,bus_type>,-
02A0                    <mailbox,mailbox_type>,-
02A0                    <journal,journal_type>,-
02A0                    <misc,misc_type>-
02A0                    >
0308
0308    disk_type:
0308            table   dt$_,<RK06,RK07,RP04,RP05,RP06,RM03,RP07,RP07HT,RL01,RL02,-
0308                    RX02,RX04,RM80,TU58,RM05,RX01,ML11,RB02,RB80,RA80,RA81,RA60,-
0308                    RZ01,RC25,RZF01,RCF25,RD51,RX50,RD52,RD53,RD26,RA82,RC26,-
0308                    RCF26,CRX50>
0428
```

```
0428     tape_type:
0428             table   dt$_,<TE16,TU45,TU77,TS11,TU78,TA78,TU80,TU81,TA81,TK50>
0480
0480     scom_type:
0480             table   dt$_,<DMC11,DMR11,XK_3271,XJ_2780,NW_X25,NV_X29,SB_ISB11,-
0480                     MX_MUX200,DMP11,DMF32,XV_3271,CI,NI,DEUNA,YN_X25,YO_X25,-
0480                     YP_ADCCP,YQ_3271,YR_DDCMP,YS_SDLC,UK_KTC32,DEQNA,DMV11,DELUA>
0548
0548     card_type:
0548             table   dt$_,<CR11>
0558
0558     term_type:
0558             table   dt$_,<TTYUNKN,VT05,FT1,FT2,FT3,FT4,FT5,FT6,FT7,FT8,LAX,-
0558                     LA36,LA120,VT5X,VT52,VT55,TQ_BTS,TEK401X,VT100,VK100,-
0558                     VT173,LA34,LA38,LA12,LA24,LQP02,VT101,VT102,VT105,VT125,-
0558                     VT131,VT132,DZ11,DZ32,DZ730,DMZ32,DHV,DHU>
0690
0690     lp_type:
0690             table   dt$_,<LP11,LA11,LA180>
06B0
06B0     workstation_type:
06B0             table   dt$_,<VS100,VS125,VS300>
06D0
06D0     realtime_type:
06D0             table   dt$_,<LPA11,DR780,DR750,DR11W,PCL11R,PCL11T,DR11C,XI_DR11C,-
06D0                     XP_PCL11B,IX_IEX11>
0728
0728     bus_type:
0728             table   dt$_,<CI780,CI750,UQPORT,UDA50,UDA50A,LESI,TU81P,RDRX,NI>
0778
0778     mailbox_type:
0778             table   dt$_,<MBX,SHRMBX,NULL>
0798
0798     journal_type:
0798             table   dt$_,<RUJNL,BIJNL,AIJNL,ATJNL,CLJNL>
07C8
07C8     misc_type:
07C8             table   dt$_,<DN11>
07D8
07D8     vcb_disk_status:
07D8             table   vcb$v_,<write_if,write_sm,homblkbad,idxhdrbad,noalloc,-
07D8                     extfid,group,system>
0820
0820     vcb_disk_status2:
0820             table   vcb$v_,<writethru,nocache,mountver,erase,nohighwater>
0850
0850     vcb_tape_status:
0850             table   vcb$v_,<partfile,logiceovs,waimouvol,wairewind,waiusrlbl,-
0850                     cancelio,mustclose,nowrite>
0898
0898     vcb_tape_mode:
0898             table   vcb$v_,<ovrexp,ovracc,ovrlbl,ovrsetid,intchg,ebcdic,novol2,-
0898                     starfile,enusereot,blank,init,noauto,ovrvolo>
0908
0908     vcb_journal_char:
0908             table   vcb$v_,<jnl_disk,jnl_tape,jnl_tmpfi>
0928
```

```
0928        cddb_status:
0928                table   cddb$v_,<snglstrm,impend,initing,reconnect,resynch,polling,-
0928                        alcls_set,noconn,rstrtwait,quorlost,dapbsy,2pbsy>
0990
0990        cddb_flags:
0990                table   mscp$v_,<cf_576,cf_shadw,cf_mlths,cf_this,cf_other,cf_misc,-
0990                        cf_attn,cf_replc>
09D8
09D8        cdrp_dutuflags:
09D8                table   cdrp$v_,<cand,canio,erlip,perm,hirt,ivcmd>
0A10
0A10        request_status:
0A10                table   irp$v_,<bufio,func,pagio,complx,virtual,chained,swapio,-
0A10                        diagbuf,physio,termio,mbxio,extend,filacp,mvirp>
0A88
0A88        io_function:
0A88                table   io$_,<nop,unload,seek,recal,erasetape,packack,spacerecord,-
0A88                        writecheck,writepblk,readpblk,available,dse,setchar,sensechar,-
0A88                        writemark,wrttmkr,writelblk,readlblk,rewindoff,setmode,rewind,-
0A88                        skipfile,skiprecord,sensemode,writeof,writevblk,readvblk,-
0A88                        access,create,deaccess,delete,modify,acpcontrol>
0B98
0B98        acp_status:
0B98                table   aqb$v_,<unique,defclass,defsys,creating>
0BC0
0BC0        aqb_acptype:
0BC0                table   aqb$k_,<undefined,f11v1,f11v2,mta,net,rem,jnl>
0C00
0C00
```

```
                            0C00   345          .sbttl  display_devbyaddr -- display UCB, etc. given its address
                            0C00   346 ;---
                            0C00   347 ;
                            0C00   348 ;        display_devbyaddr
                            0C00   349 ;
                            0C00   350 ;        This routine takes the address value in TPA$L_NUMBER(AP),
                            0C00   351 ;        attempt to use it as a UCB address, and do a SHOW DEVICE
                            0C00   352 ;        for that UCB.  This is the primary support routine for
                            0C00   353 ;        the SHOW DEVICE/ADDR command.
                            0C00   354 ;
                            0C00   355 ;    Inputs:
                            0C00   356 ;
                            0C00   357 ;        AP = pointer to TPARSE block
                            0C00   358 ;
                            0C00   359 ;    Outputs:
                            0C00   360 ;
                            0C00   361 ;        The i/o data structures for that device are shown.
                            0C00   362 ;
                            0C00   363 ;---
                            0C00   364
                            0C00   365          .enable lsb
                            0C00   366
                      0DFC  0C00   367          .entry  display_devbyaddr, -
                            0C02   368                  ^m<r2,r3,r4,r5,r6,r7,r8,r8,r10,r11>
                            0C02   369
                            0C02   370          subhd   <I/O data structures>
57   000000F9'EF     9E     0C0F   371          movab   ucb, r7                         ; get local UCB home
       52   1C AC    D0     0C16   372          movl    tpa$l_number(ap), r2            ; get supposed UCB address
             136C    30     0C1A   373          bsbw    get_ucb                         ; pull UCB to local memory
            06 50    E9     0C1D   374          blbc    r0, 900$                        ; if error, exit
     0A A7     10    91     0C20   375          cmpb    #dyn$c_ucb, ucb$b_type(r7)      ; is it really a UCB?
               4E    13     0C24   376          beql    10$                             ; branch if really a UCB
            1C AC    DD     0C26   377 900$:    pushl   tpa$l_number(ap)                ; else, output a error
                            0C29   378          type    1,<!XL is not the address of a UCB>
             006D    31     0C71   379          brw     999$                            ; then exit
                            0C74   380
56   00000071'EF     9E     0C74   381 10$:     movab   ddb, r6                         ; get local DDB home
                            0C7B   382          trymem  @ucb$l_ddb(r7), (r6), #ddb$k_length ; copy the DDB
            96 50    E9     0C8D   383 910$:    blbc    r0, 900$                        ; quit now, if error
     0A A6     06    91     0C90   384          cmpb    #dyn$c_ddb, ddb$b_type(r6)      ; is this a DDB?
               90    12     0C94   385 911$:    bneq    900$                            ; branch if not a DDB
5B   00000000'EF     9E     0C96   386          movab   sb, r11                         ; get local SB home
                            0C9D   387          trymem  @ddb$l_sb(r6), (r11), #sb$k_length ; copy the SB
            DB 50    E9     0CAF   388          blbc    r0, 910$                        ; if error, exit
 0A AB   0760 8F     B1     0CB2   389          cmpw    #<dyn$c_scs_sb@8+dyn$c_scs>, -  ; is this really a SB?
                            0CB8   390                  sb$b_type(r11)
               DA    12     0CB8   391          bneq    911$                            ; branch if no really a SB
                            0CBA   392
   10 38 A7    0E    E1     0CBA   393          bbc     #dev$v_fod, ucb$l_devchar(r7), - ; branch if this device not
                            0CBF   394                  27$                             ;    file oriented?
      50    44 AB    9A     0CBF   395          movzbl  sb$t_nodename(r11), r0          ; else, get node name size
               0D    13     0CC3   396          beql    30$                             ; branch if no node name
    45 AB40    24    90     0CC5   397          movb    #^a/$/, -                       ; add "$" to node  name
                            0CCA   398                  sb$t_nodename+1(r11)[r0]
            44 AB    96     0CCA   399          incb    sb$t_nodename(r11)              ; increase size of node name
               03    11     0CCD   400          brb     30$
            44 AB    94     0CCF   401 27$:     clrb    sb$t_nodename(r11)              ; non-fod devices have no node
```

```
                              0CD2    402
             00      DD       0CD2    403 30$:     pushl   #0                          ; setup no flags flags longword
          44 AB      9F       0CD4    404          pushab  sb$t_nodename(r11)          ; setup node name
             52      DD       0CD7    405          pushl   r2                          ; setup UCB VA
       7E    56      7D       0CD9    406          movq    r6, -(sp)                   ; setup local DDB and UCB
 1C7B'CF     05      FB       0CDC    407          calls   #5, w^show_ucb              ; display this UCB
                              0CE1    408
                     04       0CE1    409 999$:    ret
                              0CE2    410
                              0CE2    411          .disable lsb
```

```
                           0CE2               .sbttl  display_device -- display i/o data structures
                           0CE2      ;---
                           0CE2      ;
                           0CE2      ;        display_device
                           0CE2      ;
                           0CE2      ;        This routine displays all i/o data structures related
                           0CE2      ;        to a specified generic device name.
                           0CE2      ;
                           0CE2      ;    Inputs:
                           0CE2      ;
                           0CE2      ;        AP = pointer to TPARSE block
                           0CE2      ;
                           0CE2      ;    Outputs:
                           0CE2      ;
                           0CE2      ;        The i/o data structures for that device are shown.
                           0CE2      ;
                           0CE2      ;---
                           0CE2               .enabl  lsb
                           0CE2
                           0CE2      display_device::
                 0FFC      0CE2               .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                           0CE4
              58   D4      0CE4               clrl    r8                      ; init internal flags
            00B4   30      0CE6               bsbw    parse_device            ; parse the device into name and unit
                           0CE9
                           0CE9               subhd   <I/O data structures>
                           0CF6
                           0CF6               assume  flag_v_one_unit eq 0
           05 58   E8      0CF6               blbs    r8, 10$                 ; if explicit unit, skip ddb info
    0DFD'CF   6C   FA      0CF9               callg   (ap),w^show_ddbs        ; show DDB summary
                           0CFE
                           0CFE               ; init iodb scan
              5B   D4      0CFE      10$:     clrl    r11                     ; make get_ddb initialize
                           0D00
                           0D00               ; loop over all DDBs and both paths
            0202   30      0D00      20$:     bsbw    get_ddb                 ; get the next DDB
           3C 50   E9      0D03               blbc    r0,45$                  ; leave when done
57   000000F9'EF   9E      0D06               movab   ucb, r7                 ; address UCB in local storage
        58    02   CA      0D0D               bicl    #flag_m_alt_path, r8    ; assume not alternate path, yet
  52     04 A6     D0      0D10               movl    ddb$l_ucb(r6), r2       ; Address of first UCB
              06   13      0D14               beql    30$                     ; Branch if none
            1270   30      0D16               bsbw    get_ucb                 ; Read first UCB
           14 50   E8      0D19               blbs    r0, 40$                 ; If got something, go process it
  52     40 A6     D0      0D1C      30$:     movl    ddb$l_dp_ucb(r6), r2    ; try looking at the alternate path
              DE   13      0D20               beql    20$                     ; branch if nothing there
            1264   30      0D22               bsbw    get_ucb                 ; read first alternate pathed UCB
           F4 50   E9      0D25               blbc    r0, 30$                 ; if nothing there, skip this DDB
        58    02   C8      0D28               bisl    #flag_m_alt_path, r8    ; now doing the alternate path
           04 A6   D5      0D2B               tstl    ddb$l_ucb(r6)           ; was anything found on primary path?
              30   12      0D2E               bneq    60$                     ; if so, skip the controller info
                           0D30
                           0D30               ; display controller information if appropriate
                           0D30               assume  flag_v_one_unit eq 0
           2D 58   E8      0D30      40$:     blbs    r8, 60$                 ; if explicit unit, skip controler info
              59   DD      0D33               pushl   r9                      ; SVA of DDB
           44 AB   9F      0D35               pushab  sb$t_nodename(r11)      ; address of nodename
        7E    56   7D      0D38               movq    r6,-(sp)                ; address of DDB,UCB blocks
```

```
   0FE1'CF    03   FB  0D3B             calls   #3,w^show_controller     ; Display controller info
              28   11  0D40             brb     70$                      ; ...enter loop
                       0D42
                       0D42      ; Intermediate branch to final cleanup/error processing.
                       0D42
              3A   11  0D42      45$:   brb     100$
                       0D44
                       0D44             ; loop over all UCBs on a either DDB chain
     09 58    01   E1  0D44      50$:   bbc     #flag_v_alt_path, r8, - ; branch if using primary chain
                       0D48                     53$
     52    00A4 C7 D0  0D48             movl    ucb$l_dp_link(r7), r2    ; else, addr. of next UCB on sec. chain
              B1   13  0D4D             beql    20$                      ; branch if no more
              06   11  0D4F             brb     55$                      ; else, continue processing
     52      30 A7 D0  0D51      53$:   movl    ucb$l_link(r7), r2       ; address of next UCB in primary chain
              C5   13  0D55             beql    30$                      ; branch if no more
            122F   30  0D57      55$:   bsbw    get_ucb                  ; Get local copy of the UCB
           BF 50   E9  0D5A             blbc    r0, 30$                  ; skip rest if chain broken
                       0D5D             assume  flag_v_one_unit eq 0
           0A 58   E9  0D5D             blbc    r8, 70$                  ; branch if displaying all units
                       0D60
00000571'EF    54 A7   B1  0D60      60$:   cmpw    ucb$w_unit(r7), -        ; check if request unit
                       0D68                     parsed_devnam+pdvnm_w_unit
              DA   12  0D68             bneq    50$                      ; skip if not
                       0D6A
              58   DD  0D6A      70$:   pushl   r8                       ; flags longword
           44 AB   9F  0D6C             pushab  sb$t_nodename(r11)       ; address of node name
              52   DD  0D6F             pushl   r2                       ; actual address of UCB
          7E  56   7D  0D71             movq    r6,-(sp)                 ; address of DDB,UCB blocks
     1C7B'CF  05   FB  0D74             calls   #5,w^show_ucb            ; display current UCB
          58  04   C8  0D79             bisl    #flag_m_fnd_unit, r8     ; mark at least 1 UCB was displayed
              C6   11  0D7C             brb     50$                      ; loop thru all UCB's
                       0D7E
          58  02   E0  0D7E     100$:   bbs     #flag_v_fnd_unit, -      ; branch if at least 1 ucb displayed
              13       0D81                     r8, 110$
     50   0000'8F  3C  0D82             movzwl  #ss$_nosuchdev,r0        ; signal "no such device"
                       0D87             signal  0
                       0D95     110$:   status  success                  ; exit to tparse w/success
                   04  0D9C             ret
                       0D9D
                       0D9D             .dsabl  lsb
```

```
                              0D9D                   .sbttl  parse_device -- parse device name into name and unit number
                              0D9D   ;---
                              0D9D   ;
                              0D9D   ;               parse the device name into name and unit number
                              0D9D   ;
                              0D9D   ; Inputs:
                              0D9D   ;
                              0D9D   ;               r8 = longword of show command status flags
                              0D9D   ;               tpa$l_tokencnt(ap) = Descriptor of device name
                              0D9D   ;               parsed_devnam = address of a work area into which parsed fragments
                              0D9D   ;                               of the device name are stored
                              0D9D   ;
                              0D9D   ; Outputs:
                              0D9D   ;
                              0D9D   ;               if x equals parsed_devnam then:
                              0D9D   ;                   pdvnm_t_node(x) = ASCIC string for parsed node name
                              0D9D   ;                   pdvnm_t_ddc(x)  = ASCIC string for parsed device and controller
                              0D9D   ;                   pdvnm_s_unit(x) = converted unit number
                              0D9D   ;                    (null strings imply item missing from input)
                              0D9D   ;               flag_m_one_unit in r8, set if unit number specified
                              0D9D   ;               r2-r7 and r9-r11 are destroyed.
                              0D9D   ;---
                              0D9D
                              0D9D   parse_device:
     5B   00000551'EF     9E  0D9D           movab   parsed_devnam, r11      ; get working area base address
                   6B     D4  0DA4           clrl    pdvnm_t_node(r11)       ; null the two string values
                10 AB     D4  0DA6           clrl    pdvnm_t_ddc(r11)
                20 AB     B4  0DA9           clrw    pdvnm_w_unit(r11)       ; zero unit number
           56   10 AC     7D  0DAC           movq    tpa$l_tokencnt(ap), r6  ; get descriptor of input string
        67   56   24      3A  0DB0           locc    #^a/$/, r6, (r7)        ; scan name for a "$"
                   14     13  0DB4           beql    10$                     ; branch if none
        59   51   57      C3  0DB6           subl3   r7, r1, r9              ; compute size of node name
     01 AB   67   59      28  0DBA           movc3   r9, (r7), -             ; copy node name string to work area
                              0DBF                   pdvnm_t_node+1(r11)
             6B   59      90  0DBF           movb    r9, pdvnm_t_node(r11)   ; store node name size
                  59      D6  0DC2           incl    r9                      ; get size of node name incl. "$"
             56   59      C2  0DC4           subl    r9, r6                  ; adjust input string descriptor to
             57   59      C0  0DC7           addl    r9, r7                  ;  remove node name section
                  56      D5  0DCA   10$:    tstl    r6                      ; anything left to work with?
                  2E      13  0DCC           beql    90$                     ; branch if no characters left
        50   67   30      B3  0DCE   20$:    subb3   #^a/0/, (r7), r0        ; convert next character to a
                  12      19  0DD2           blss    50$                     ;  a numeric value and branch to
             09   50      91  0DD4           cmpb    r0, #9                  ;  50$ if not a numeric digit
                  0D      1A  0DD7           bgtru   50$
          20 AB   0A      A4  0DD9           mulw    #10, pdvnm_w_unit(r11)  ; scale unit number by ten
          20 AB   50      A0  0DDD           addw    r0, pdvnm_w_unit(r11)   ;  and add new digit
             58   01      C8  0DE1           bisl    #flag_m_one_unit, r8    ; set the unit number found flag
                  11      11  0DE4           brb     66$                     ; go do next digit
                              0DE6   50$:    assume  flag_v_one_unit eq 0
                13 58     E8  0DE6           blbs    r8, 90$                 ; branch if unit number already found
           50   10 AB     9A  0DE9           movzbl  pdvnm_t_ddc(r11), r0    ; get number of characters in dev/ctrl
       11 AB40    67      90  0DED           movb    (r7), -                 ; move new character into place
                              0DF2                   pdvnm_t_ddc+1(r11)[r0]
     10 AB   50   01      81  0DF2           addb3   #1, r0, pdvnm_t_ddc(r11) ; store new character count
                  57      D6  0DF7   66$:    incl    r7                      ; move string pointer
                D2 56     F5  0DF9           sobgtr  r6, 20$                 ; reduce character count and branch
                              0DFC                                           ;  if characters still left to process
                          05  0DFC   90$:    rsb
```

```
                                      0DFD            .sbttl  show_ddbs -- display device data blocks (DDBs)
                                      0DFD     ;---
                                      0DFD     ;
                                      0DFD     ;       show_ddbs
                                      0DFD     ;
                                      0DFD     ;       This routine displays all active DDB's associated
                                      0DFD     ;       with a specified generic device name.
                                      0DFD     ;
                                      0DFD     ;   Inputs:
                                      0DFD     ;
                                      0DFD     ;       AP = pointer to TPARSE block
                                      0DFD     ;
                                      0DFD     ;---
                                      0DFD            .save
                                  000008D2            .psect literals
                                      08D2
                                      08D2     found_dpt:
                   000008DA'00000008' 08D2            .address 8, 10$
                                      08DA     10$:   string  <!_!XL    !10<!AC!AC!>    !6AD!+!+    !10AC !XL  !XW>
                                      0915
                                      0915     no_dpt:
                   0000091D'00000006' 0915            .address 6, 10$
                                      091D     10$:   string  <!_!XL    !10<!AC!AC!>    !6AD!+!+    !10AC>
                                      094F
                                  00000DFD            .restore
                                      0DFD
                                      0DFD     show_ddbs:
                                OFFC  0DFD            .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                                      0DFF
                                      0DFF            skip    page
                                      0E06            print   0,<!_!_!_!_DDB list>
                                      0E13            print   0,<!_!_!_!_--------->
                                      0E20            skip    1
                                      0E29            print   0,<!_ Address    Controller    ACP       Driver      DPT   DPT size
                                      0E36            print   0,<!_ -------    ----------    ---       ------      ---   --------
                                      0E43            skip    1
                           5B   D4    0E4C            clrl    r11                        ; make get_ddb initialize
                                      0E4E
                         00B4   30    0E4E     10$:   bsbw    get_ddb                    ; find next DDB
                         62 50  E9    0E51            blbc    r0, 90$                    ; end of DDB list
            54    00000915'EF   7D    0E54            movq    no_dpt, r4                 ; assume no DPT will be found
                           5A   10    0E5B            bsbb    find_dpt                   ; locate dpt; r7 = local dpt; r8 = address
                         0D 50  E9    0E5D            blbc    r0, 17$                    ; branch if not found
            54    000008D2'EF   7D    0E60            movq    found_dpt, r4              ; show that DPT was found
                    7E   08 A7  3C    0E67            movzwl  dpt$w_size(r7), -(sp)      ; length of DPT
                           58   DD    0E6B            pushl   r8                         ; address of DPT
                        24 A6   DF    0E6D     17$:   pushal  ddb$t_drvname(r6)          ; address of driver name
                           7E   7C    0E70            clrq    -(sp)                      ; allocate 2 longwords for ACP name
                           6E   DF    0E72            pushal  (sp)                       
                           7E   D4    0E74            clrl    -(sp)                      ; assume no ACP name for this DDB
    50    10 A6   FF000000 8F   CB    0E76            bicl3   #^xff000000, -             ; obtain ACP name for this DDB
                                      0E7F                    ddb$l_acpd(r6), r0
                           20   13    0E7F            beql    30$                        ; branch if no ACP name in this DDB
                   08 AE   50   D0    0E81            movl    r0, 8(sp)                  ; put name in the working string
                      6E   06   D0    0E85            movl    #6, (sp)                   ; set length of ACP name
         0B AE    00505158 8F   D0    0E88            movl    #^a'XQP', 11(sp)           ; assume ACP is really an XQP
            50    00313146 8F   D1    0E90            cmpl    #^a'F11', r0               ; is it an XQP?
```

```
                           08  13  0E97   626         beql    30$                     ; branch if its an XQP
      0B AE    00504341 8F  D0  0E99   627         movl    #^a'ACP', 11(sp)        ; else, change it to an ACP
                     14 A6  DF  0EA1   628 30$:    pushal  ddb$t_name(r6)          ; generic device name for controller
                     44 AB  9F  0EA4   629         pushab  sb$t_nodename(r11)      ; node name
                        59  DD  0EA7   630         pushl   r9                      ; actual address of DDB
                                0EA9   631         printd  r4, (r5)                ; print a line
                        98  11  0EB4   632         brb     10$                     ; loop till out of DDBs
                            04  0EB6   633 90$:    ret                             ; then return
                            0EB7   635 ;
                            0EB7   636 ;       Subroutine to locate the DPT corresponding to the current
                            0EB7   637 ;       DDB.
                            0EB7   638 ;
                            0EB7   639 find_dpt:
                  3C    BB  0EB7   640         pushr   #^m<r2,r3,r4,r5>
   57   00000439'EF     9E  0EB9   641         movab   dpt,r7
                            0EC0   642         trymem  @ioc$gl_dptlist,dpt$l_flink(r7) ; set address of first DPT
               2F 50    E9  0ED0   643         blbc    r0,90$                  ; branch if error
              58  67    D0  0ED3   644 10$:    movl    dpt$l_flink(r7),r8      ; skip to next DPT
   00000000'EF    58    D1  0ED6   645         cmpl    r8,ioc$gl_dptlist       ; check if back to listhead
                  21    13  0EDD   646         beql    80$                     ; branch if end of list
                            0EDF   647         trymem  (r8),(r7),#dpt$c_length ; read the entire dpt
               13 50    E9  0EEC   648         blbc    r0,90$                  ; branch if error
          50   20 A7    9A  0EEF   649         movzbl  dpt$t_name(r7),r0       ; get length of dpt driver name
 25 A6    21 A7   50    29  0EF3   650         cmpc    r0,dpt$t_name+1(r7),ddb$t_drvname+1(r6)
                  D8    12  0EF9   651         bneq    10$                     ; branch if no match yet
           50     01    D0  0EFB   652 50$:    movl    #1,r0                   ; success
                  02    11  0EFE   653         brb     90$
                  50    D4  0F00   654 80$:    clrl    r0                      ; not found
                  3C    BA  0F02   655 90$:    popr    #^m<r2,r3,r4,r5>
                  05        0F04   656         rsb
```

```
                                 0F05         .sbttl  get_ddb -- locate the next DDB in the I/O database
                                 0F05 ;---
                                 0F05 ;
                                 0F05 ;       get_ddb
                                 0F05 ;
                                 0F05 ;       This routine locates the next DDB in the I/O database.  All
                                 0F05 ;       available system blocks are searched.  However, if a node name
                                 0F05 ;       is specified, only the system block whose node name matches
                                 0F05 ;       actually has DDBs returned.
                                 0F05 ;
                                 0F05 ; Inputs:
                                 0F05 ;
                                 0F05 ;       r6 -    addr of DDB, local storage
                                 0F05 ;       r11 -   addr of SB, local storage
                                 0F05 ;               (zero means initialize scan)
                                 0F05 ;
                                 0F05 ; Outputs:
                                 0F05 ;
                                 0F05 ;       r0 -    status
                                 0F05 ;       r6 -    addr of DDB, local storage
                                 0F05 ;       r9 -    SYS VA of DDB
                                 0F05 ;       r11-    addr of SB, local storage
                                 0F05 ;
                                 0F05 ;---
                                 0F05
                                 0F05 get_ddb:
                       5B    D5  0F05         tstl    r11                     ; must we initialize?
                       64    13  0F07         beql    1500$                   ; branch if must initialize
                                 0F09
                 59    66    D0  0F09 10$:    movl    ddb$l_link(r6),r9       ; skip to next DDB
                       61    13  0F0C         beql    100$                    ; if end of list, go try next SB
                                 0F0E         getmem  (r9), (r6), -           ; read entire DDB
                                 0F0E                 #ddb$c_length
                    4A 50    E9  0F1F         blbc    r0, 90$                 ; skip if cannot read
     57   00000551'EF    9E  0F22         movab   parsed_devnam, r7       ; get parsed device name data base addr.
           51     10 A7    9A  0F29         movzbl  pdvnm_t_ddc(r7), r1     ; was generic device specified?
                       0E    13  0F2D         beql    50$                     ; branch if not
              14 A6    51    91  0F2F         cmpb    r1, ddb$t_name(r6)      ; Is device name big enough?
                       D4    1A  0F33         bgtru   10$                     ; branch if not
  15 A6     11 A7    51    29  0F35         cmpc3   r1, pdvnm_t_ddc+1(r7), -
                                 0F3B                 ddb$t_name+1(r6)
                       CC    12  0F3B         bneq    10$                     ; loop until end of list
          44 AB     22 A7    90  0F3D 50$:    movb    pdvnm_b_nodesz(r7), -   ; assume that the node name is
                                 0F42                 sb$t_nodename(r11)      ;  required for this DDB
 00000000'EF     34 A6    D1  0F42         cmpl    ddb$l_sb(r6), -         ; is this the local node?
                                 0F4A                 scs$ga_localsb
                       1D    12  0F4A         bneq    70$                     ; no, node name is required
              51     04 A6    D0  0F4C         movl    ddb$l_ucb(r6), r1       ; for the local node, we want to
                       06    12  0F50         bneq    53$                     ;  show a node name if and only if
              51     40 A6    D0  0F52         movl    ddb$l_dp_ucb(r6), r1    ;  this is a file oriented device
                       11    13  0F56         beql    70$                     ; if we cannot tell, show the node name
                                 0F58 53$:    getmem  ucb$l_devchar(r1)       ; else test for a file oriented device
              03 51    0E    E0  0F62         bbs     #dev$v_fod, r1, 70$     ;  using device characteristics flag
                    44 AB    94  0F66         clrb    sb$t_nodename(r11)      ; if not fod, vanish node name
                 50    01    D0  0F69 70$:    movl    #1,r0                   ; set success
                       05  0F6C 90$:    rsb
                       52    11  0F6D 1500$:  brb     500$                    ; branch assist
```

```
                                  0F6F
                                  0F6F  ;
                                  0F6F  ; move to next SB
                                  0F6F  ;
                                  0F6F
                    50      D4    0F6F  100$:   clrl    r0                        ; Set for failure
               5A   6B      D0    0F71          movl    sb$l_flink(r11), r10      ; Get next block
      00000000'EF   5A      D1    0F74          cmpl    r10, scs$gq_config        ; Reached end of queue?
                    EF      13    0F7B          beql    90$                       ; yes
                                  0F7D          getmem  (r10), (r11), -           ; Pick up system block
                                  0F7D                  #sb$c_length
                 DB 50      E9    0F8E          blbc    r0,90$                    ; exit if broken
                 54 AB      D0    0F91          movl    sb$l_ddb(r11),-
                    66            0F94                   ddb$l_link(r6)           ; set address of first DDB
  5A   00000551'EF          9E    0F95          movab   parsed_devnam, r10        ; get parsed device name data base addr.
          50     44 AB      9A    0F9C          movzbl  sb$t_nodename(r11), r0    ; get size of node name
                    0A      13    0FA0          beql    120$                      ; branch if no node name
        45 AB40     24      90    0FA2          movb    #^a/$/, -                 ; append "$" to the node name
                                  0FA7                  sb$t_nodename+1(r11)[r0]
    22 AA     50    01      81    0FA7          addb3   #1, r0, pdvnm_b_nodesz(r10) ; store new node name size
              55    6A      9A    0FAC  120$:   movzbl  pdvnm_t_node(r10), r5     ; pick up requested node name lenght
                    0D      13    0FAF          beql    130$                      ; there is none, go scan DDB chain
              50    55      91    0FB1          cmpb    r5, r0                    ; do length match?
                    B9      12    0FB4          bneq    100$                      ; no, this cannot be it
  45 AB    01 AA    55      29    0FB6          cmpc3   r5, -                     ; do names match?
                                  0FBC                  pdvnm_t_node+1(r10), -
                                  0FBC                  sb$t_nodename+1(r11)
                    B1      12    0FBC          bneq    100$                      ; no, this cannot be it
                  FF48      31    0FBE  130$:   brw     10$                       ; go scan the DDB chain
                                  0FC1
                                  0FC1  ;
                                  0FC1  ; initialize I/O database scan
                                  0FC1  ;
                                  0FC1
  5B   00000000'EF          9E    0FC1  500$:   movab   sb, r11                   ; pickup local SB storage address
  56   00000071'EF          9E    0FC8          movab   ddb, r6                   ; pickup local DDB storage address
                                  0FCF          getmem  @scs$gq_config, -         ; initialize next SB pointer
                                  0FCF                  sb$l_flink(r11)
                    BE      11    0FDF          brb     100$                      ; link to next SB
```

```
                            0FE1          .sbttl  show_controller, Display controller information
                            0FE1  ;---
                            0FE1  ;
                            0FE1  ;       show_controller
                            0FE1  ;
                            0FE1  ;       Display all information related to the controller
                            0FE1  ;       device associated with each generic device name.
                            0FE1  ;
                            0FE1  ;   Inputs:
                            0FE1  ;
                            0FE1  ;       4(ap) = Address of DDB in local storage
                            0FE1  ;       8(ap) = Address of UCB in local storage
                            0FE1  ;       12(ap)= Address of node name in local storage
                            0FE1  ;       16(ap)= SVA of DDB
                            0FE1  ;
                            0FE1  ;---
                            0FE1
                            0FE1  show_controller:
                      00FC  0FE1          .word   ^m<r2,r3,r4,r5,r6,r7>
            52  04 AC  7D   0FE3          movq    4(ap),r2                ; get address of DDB,UCB
   54   00000000'EF    9E   0FE7          movab   buffer,r4
                            0FEE
                            0FEE  ; begin with controller heading
                            0FEE
                            0FEE          skip    page
                14 A2  DF   0FF5          pushal  ddb$t_name(r2)          ; generic controller name
                0C AC  DD   0FF8          pushl   12(ap)
                            0FFB          print   2,<Controller: !AC!AC>
                   0C  DD   1008          pushl   #12
            6E  14 A2  80   100A          addb    ddb$t_name(r2), (sp)
            6E  0C BC  80   100E          addb    @12(ap), (sp)
                            1012          print   1,<!#*->
                            101F          skip    1
                            1028
 00000000'EF    34 A2  91   1028          cmpb    ddb$l_sb(r2), scs$ga_localsb          ; skip this stuff if
                   08  13   1030          beql    skip_sb                               ;  this is the local SB
                34 A2  DD   1032          pushl   ddb$l_sb(r2)                          ; else, display SB and
        17D8'CF    01  FB   1035          calls   #1, w^show_system_block               ;  related information
                            103A
                            103A  skip_sb:
                            103A          getmem  @16(ap), (r4), #ddb$k_length          ; copy DDB to local mem.
                            104C          retiferr
                            1050          ensure  6
                10 AC  DD   1068          pushl   16(ap)
                            106B          print   1,<!_!_--- Device Data Block (DDB) !XL --->
                            1078          skip    1
                            1081          print_columns -
                            1081                  buffer, 16(ap), -
                            1081                  ddb_column_1, ddb_column_2, ddb_column_3
                            10A3          skip    1
                            10AC
                            10AC          getmem  @ucb$l_crb(r3), (r4), #crb$k_length     ; get primary CRB
                            10BE          retiferr
                            10C2          ensure  8
                24 A3  DD   10DA          pushl   ucb$l_crb(r3)
                            10DD          print   1,<!_    ---  Primary Channel Request Block (CRB) !XL --->
                            10EA          skip    1
```

```
      00000578'EF    40 A3    90  10F3    812            movb    ucb$b_devclass(r3), crb_devclass          ; setup device info.
                                  10FB    813            print_columns -
                                  10FB    814                    buffer, ucb$l_crb(r3), -                  ; output CRB columns
                                  10FB    815                    crb_column_1, crb_column_2, crb_column_3
           50    24 A3    24   C1 111D    816            addl3   #crb$l_intd, ucb$l_crb(r3), r0
                                  1122    817            print_columns -
                                  1122    818                    buffer+crb$l_intd, r0, -                  ; output VEC columns
                                  1122    819                    vec_column_1, vec_column_2, vec_column_3
                                  1143    820            skip    1
                                  114C    821
                57    20 A4    D0  114C    822            movl    crb$l_link(r4), r7                        ; link to second. CRB
                         03    12  1150    823            bneq    10$
                       0093    31  1152    824            brw     skip_second_crb                           ; branch if none
                                  1155    825 10$:       getmem  (r7), (r4), #crb$k_length                 ; get secondary CRB
                                  1166    826            retiferr
                                  116A    827            ensure  8
                         57    DD  1182    828            pushl   r7
                                  1184    829            print   1,<!_  ---  Secondary Channel Request Block (CRB) !XL --->
                                  1191    830            skip    1
                                  119A    831            print_columns -
                                  119A    832                    buffer, r7, -                             ; output CRB columns
                                  119A    833                    crb_column_1, crb_column_2, crb_column_3
                   57    24    C0  11BB    834            addl2   #crb$l_intd, r7
                                  11BE    835            print_columns -
                                  11BE    836                    buffer+crb$l_intd, r7, -                  ; output VEC columns
                                  11BE    837                    vec_column_1, vec_column_2, vec_column_3
                                  11DF    838            skip    1
                                  11E8    839
                                  11E8    840 skip_second_crb:
      00000000'EF    34 A2    D1  11E8    841            cmpl    ddb$l_sb(r2), scs$ga_localsb              ; is this a local dev.?
                         03    13  11F0    842            beql    10$
                       0080    31  11F2    843            brw     display_ddt                               ; if so, skip IDB etc.
           57    24 A3    2C   C1 11F5    844 10$:       addl3   #<crb$l_intd+vec$l_idb>, -                ; locate address of
                                  11FA    845                    ucb$l_crb(r3), r7                         ;  primary IDB
                                  11FA    846            getmem  (r7)                                      ; get that address
                                  1203    847            retiferr
                   57    51    D0  1207    848            movl    r1, r7                                    ; save IDB address
                                  120A    849            getmem  (r7), (r4), #idb$k_length                 ; copy IDB to local mem.
                                  1217    850            retiferr
                                  121B    851            ensure  4
                         57    DD  1233    852            pushl   r7
                                  1235    853            print   1,<!_!_--- Interrupt Data Block (IDB) !XL --->
                                  1242    854            skip    1
                                  124B    855            print_columns -
                                  124B    856                    buffer, r7, -
                                  124B    857                    idb_column_1, idb_column_2, idb_column_3
                                  126C    858            skip    1
                                  1275    859
                                  1275    860 display_ddt:
                                  1275    861            getmem  @ucb$l_ddt(r3), (r4), #ddt$k_length      ; copy DDT to local mem.
                                  1284    862            retiferr
                                  1288    863            ensure  6
                    0088 C3    DD  12A0    864            pushl   ucb$l_ddt(r3)
                                  12A4    865            print   1,<!_!_--- Driver Dispatch Table (DDT) !XL --->
                                  12B1    866            skip    1
                                  12BA    867            print_columns -
                                  12BA    868                    buffer, ucb$l_ddt(r3), -
```

```
         12BA    869                        ddt_column_1, ddt_column_2, ddt_column_3
         12DD    870
    04   12DD    871           ret
```

```
                          12DE             .sbttl show_controller tables & action routines
                          12DE
                          12DE    ;  The following are all PRINT_COLUMNS action routines for the show
                          12DE    ;  controller displays.
                          12DE    ;
                          12DE    ;    Action Routine Inputs:
                          12DE    ;
                          12DE    ;       R2              value from the COLUMN_LIST entry
                          12DE    ;       R5              size of value section for this item
                          12DE    ;       R7              address of a descriptor for a scratch string in
                          12DE    ;                       which the FAO converted value is to be returned
                          12DE    ;       R11             base address of the local UCB copy
                          12DE    ;
                          12DE    ;    Action Routine Outputs:
                          12DE    ;
                          12DE    ;       R0              status
                          12DE    ;                          lbs ==> use this entry
                          12DE    ;                          lbc ==> skip this entry
                          12DE    ;       R1 - R5         scratch
                          12DE    ;                       all other registers must be preserved
                          12DE
                          12DE    ;
                          12DE    ; FAO control strings, etc. used by the action routines
                          12DE    ;
                          12DE
                          12DE             .save
                      00000B47             .psect  literals
                          0B47
                          0B47    vec_fao_datapath:
                          0B47             string  <!UB!AC!AC>
                          0B58
                          0B58    vec_fao_mapreg:
                          0B58             string  <!UB(!UB)!AC>
                          0B6B
                          0B6B    vec_lwae:
       45 41 57 4C 20 00' 0B6B             .ascic  / LWAE/
                       05 0B6B
                          0B71
                          0B71    vec_locked:
 64 65 6B 63 6F 4C 20 00' 0B71             .ascic  / Locked/
                       07 0B71
                          0B79
                          0B79    ddt_return:
    6E 72 75 74 65 72 00' 0B79             .ascic  /return/
                       06 0B79
                          0B80
                      000012DE             .restore
                          12DE
                          12DE    ;
                          12DE    ; PRINT_COLUMNS tables for DDB display
                          12DE    ;
                          12DE
                          12DE    ddb_column_1:
                          12DE             column_list -
                          12DE                     ddb$, 20, 8, 3, <-
                          12DE                     <<Driver name>,t_drvname,ac,13,15>, -
                          12DE                     <<ACP ident>,ddb_acpd,0,25,3>, -
```

```
                                  12DE                          <<ACP class>,ddb_acpcls,0>, -
                                  12DE                          >
                                  131E
                                  131E  ddb_column_2:
                                  131E          column_list -
                                  131E                          ddb$, 15, 8, 3, <-
                                  131E                          <<Alloc. class>,l_allocls,ub>, -
                                  131E                          <<SB address>,l_sb,xl>, -
                                  131E                          <<UCB address>,l_ucb,xl>, -
                                  131E                          >
                                  135E
                                  135E  ddb_column_3:
                                  135E          column_list -
                                  135E                          ddb$, 15, 8, 0, <-
                                  135E                          <<DDT address>,l_ddt,xl>, -
                                  135E                          <<CONLINK addr.>,l_conlink,xl_neq>, -
                                  135E                          <<2p UCB addr.>,l_dp_ucb,xl_neq>, -
                                  135E                          >
                                  139E
                                  139E  ;*********
                                  139E  ddb_acpd:
 52   10 AB   FF000000 8F     CB  139E          bicl3   #^xff000000, ddb$l_acpd(r11), - ; get ACP descriptor
                                  13A7                  r2
                     18       13  13A7          beql    ddb_no_acp                      ; branch if no ACP info
           52   52   08       9C  13A9          rotl    #8, r2, r2                      ; make ACP descriptor into
                52   03       C0  13AD          addl    #3, r2                          ;  an ASCIC string and
                     52       DD  13B0          pushl   r2                              ;  push it onto the stack
                52   5E       D0  13B2          movl    sp, r2                          ; save ASCIC pointer
                                  13B5          do_column_entry ac                      ; display ACP type id
                     8E       D5  13BE          tstl    (sp)+                           ; cleanup stack
                              05  13C0          rsb
                                  13C1  ddb_no_acp:
                     50       D4  13C1          clrl    r0
                              05  13C3          rsb
                                  13C4
                                  13C4  ;***********
                                  13C4  ddb_acpcls:
           52    13 AB        9A  13C4          movzbl  ddb$b_acpclass(r11), r2         ; get ACP class
                     F7       13  13C8          beql    ddb_no_acp                      ; branch if none
           53   ECC2 CF       9E  13CA          movab   ddb_acpclass, r3                ; get translate table
             00000000'GF      16  13CF          jsb     g^translate_address             ; translate ACP class
                     0C       13  13D5          beql    90$                             ; branch if translate failed
                52   50       D0  13D7          movl    r0, r2                          ; setup translated string
                                  13DA          do_column_entry ac, jmp                 ; display translation
                                  13E3
           52    13 AB        9E  13E3  90$:    movab   ddb$b_acpclass(r11), r2         ; else, get class address
                                  13E7          do_column_entry ub, jmp                 ;  just display the value
                                  13F0
                                  13F0  ;
                                  13F0  ; PRINT_COLUMNS tables for CRB display
                                  13F0  ;
                                  13F0
                                  13F0  crb_column_1:
                                  13F0          column_list -
                                  13F0                          crb$, 16, 8, 4, <-
                                  13F0                          <<Reference count>,w_refc,uw>, -
                                  13F0                          <<Due time>,crb_timeout,crb$l_duetime>, -
                              13F0   984                      >
                              1420   985
                              1420   986 crb_column_2:
                              1420   987         column_list -
                              1420   988                 crb$, 16, 8, 4, <-
                              1420   989                 <<Wait queue>,l_wqfl,q2>, -
                              1420   990                 <<Timeout rout.>,crb_timeout,crb$l_toutrout>, -
                              1420   991                 >
                              1450   992
                              1450   993 crb_column_3:
                              1450   994         column_list -
                              1450   995                 crb$, 16, 8, 0, <-
                              1450   996                 <<Aux. struct.>,l_auxstruc,xl_neq>, -
                              1450   997                 <<Timeout link>,crb_timeout,crb$l_timelink>, -
                              1450   998                 >
                              1480   999
                              1480  1000 ;***********
                              1480  1001 crb_timeout:
 00000578'EF    42 8F     91  1480  1002         cmpb    #dc$_term, -                    ; terminals have a different
                              1488  1003                 crb_devclass                    ;  timeout scheme
                11    13      1488  1004         beql    90$                             ;  so don't do them
             1C AB    D5      148A  1005         tstl    crb$l_toutrout(r11)             ; also don't bother unless
                0C    13      148D  1006         beql    90$                             ;  a time out routine specified
          52    5B    C0      148F  1007         addl    r11, r2                         ; get datum address
                              1492  1008         do_column_entry xl, jmp                 ;  and display it
                50    D4      149B  1009 90$:    clrl    r0                              ; or don't show anything
                      05      149D  1010         rsb
                              149E  1011
                              149E  1012         .save
                          00000578  1013         .psect  sdadata,noexe,wrt
                              0578  1014 crb_devclass:
               00000000       0578  1015         .long   0
                          0000149E  1016         .restore
                              149E  1017
                              149E  1018 ;
                              149E  1019 ; PRINT_COLUMNS tables for VEC display
                              149E  1020 ;
                              149E  1021
                              149E  1022 vec_column_1:
                              149E  1023         column_list -
                              149E  1024                 vec$, 16, 8, 4, <-
                              149E  1025                 <<IDB address>,l_idb,xl>, -
                              149E  1026                 <<ADP address>,l_adp,xl_neq>, -
                              149E  1027                 <<Unit start rout.>,l_start,xl_neq>, -
                              149E  1028                 >
                              14DE  1029
                              14DE  1030 vec_column_2:
                              14DE  1031         column_list -
                              14DE  1032                 vec$, 16, 8, 4, <-
                              14DE  1033                 <<Datapath>,vec_datapath,0,10,14>, -
                              14DE  1034                 <<Unit init.>,l_unitinit,xl_neq>, -
                              14DE  1035                 <<Disc. rout.>,l_unitdisc,xl_neq>, -
                              14DE  1036                 >
                              151E  1037
               00000004       151E  1038 vec$l_intser = vec$q_dispatch+4
                              151E  1039 vec_column_3:
                              151E  1040         column_list -
                              151E                         vec$, 16, 8, 0, <-
                              151E                         <<Map reg.>,vec_mapreg,0,11,13>, -
                              151E                         <<Int. service>,l_intser,xl_neq>, -
                              151E                         <<Ctrl. init.>,l_initial,xl_neq>, -
                              151E                         >
                              155E
                              155E      ;************
                              155E      vec_datapath:
                  5E   10     155E               bsbb      vec_test_uba                   ; is this a UNIBUS?
               5E   18   C2   1560               subl      #<8+16>, sp                    ; make scratch space on stack
               52   5E   D0   1563               movl      sp, r2                         ; point to string descriptor
               62   10   D0   1566               movl      #16, (r2)                      ; build string descriptor
         04 A2   08 A2   9E   1569               movab     8(r2), 4(r2)
      53  00000E21'EF    9E   156E               movab     null_ascic, r3                 ; assume no LWAE
         07 13 AB   05   E1   1575               bbc       #vec$v_lwae, -                 ; branch if LWAE not on
                              157A                         vec$b_datapath(r11), 10$
      53  00000B6B'EF    9E   157A               movab     vec_lwae, r3                   ; else, change assumption
      54  00000E21'EF    9E   1581      10$:     movab     null_ascic, r4                 ; assume no pathlock
         07 13 AB   07   E1   1588               bbc       #vec$v_pathlock, -             ; branch if path not locked
                              158D                         vec$b_datapath(r11), 20$
      54  00000B71'EF    9E   158D               movab     vec_locked, r4                 ; else, change assumption
 51    13 AB   05   00   EF   1594      20$:     extzv     #vec$v_datapath, -             ; extract data path number
                              159A                         #vec$s_datapath, -
                              159A                         vec$b_datapath(r11), r1
                              159A               $fao_s    -
                              159A                         ctrstr = vec_fao_datapath, -   ; convert everything to
                              159A                         outbuf = (r2), -               ;  to a string
                              159A                         outlen = (r2), -
                              159A                         p1 = r1, -
                              159A                         p2 = r3, -
                              159A                         p3 = r4
                              15B1               do_column_entry as                       ; put string in column
               5E   18   C0   15BA               addl      #<8+16>, sp                    ; cleanup stack
                         05   15BD               rsb
                              15BE
                              15BE
                              15BE      ;************
                              15BE      vec_test_uba:
            50   14 AB   D0   15BE               movl      vec$l_adp(r11), r0             ; get ADP address
                    13   13   15C2               beql      90$                            ; if none, its not a UBA
                              15C4               getmem    adp$w_adptype(r0)              ; get adapter type
                 06 50   E9   15CE               blbc      r0, 90$                        ; if error, its not a UBA
               51   01   B1   15D1               cmpw      #at$_uba, r1                   ; is it a UBA?
                    01   12   15D4               bneq      90$                            ; branch if not a UBA
                         05   15D6               rsb                                      ; else, return to caller
                    8E   D5   15D7      90$:     tstl      (sp)+                          ; if not a UBA, return a skip
                    50   D4   15D9               clrl      r0                             ;  this entry status to the
                         05   15DB               rsb                                      ;  action routines caller
                              15DC
                              15DC      ;**********
                              15DC      vec_mapreg:
                    E0   10   15DC               bsbb      vec_test_uba                   ; is this a UBA?
               5E   18   C2   15DE               subl      #<8+16>, sp                    ; make scratch space on stack
               52   5E   D0   15E1               movl      sp, r2                         ; point to string descriptor
               62   10   D0   15E4               movl      #16, (r2)                      ; build string descriptor
         04 A2   08 A2   9E   15E7               movab     8(r2), 4(r2)
      54  00000E21'EF    9E   15EC               movab     null_ascic, r4                 ; assume no map lock
```

```
        07 10 AB    0F     E1  15F3                bbc      #vec$v_maplock, -                  ; branch if no map lock
                               15F8                         vec$w_mapreg(r11), 10$
      54   00000B71'EF     9E  15F8                movab    vec_locked, r4                     ; else, change assumption
  53   10 AB    0F   00    EF  15FF      10$:      extzv    #vec$v_mapreg, #vec$s_mapreg, -   ; extract starting map
                               1605                         vec$w_mapreg(r11), r3              ;  number
                               1605                $fao_s   -
                               1605                         ctrstr = vec_fao_mapreg, -        ; convert whole mess to a
                               1605                         outbuf = (r2), -                   ;  string
                               1605                         outlen = (r2), -
                               1605                         p1 = r3, -
                               1605                         p2 = vec$b_numreg(r11), -
                               1605                         p3 = r4
                               161D                do_column_entry as                          ; put string in column
              5E    18     C0  1626                addl     #<8+16>, sp                        ; cleanup stack
                           05  1629                rsb
                               162A
                               162A      ;
                               162A      ; PRINT_COLUMNS tables for IDB display
                               162A      ;
                               162A
                               162A      idb_column_1:
                               162A                column_list -
                               162A                         idb$, 16, 8, 4, <-
                               162A                         <<CSR address>,l_csr,xl>, -
                               162A                         <<Number of units>,w_units,uw>, -
                               162A                         >
                               165A
                               165A      idb_column_2:
                               165A                column_list -
                               165A                         idb$, 16, 8, 4, <-
                               165A                         <<Owner UCB addr.>,l_owner,xl>, -
                               165A                         <<Interrupt vector>,idb_vector,0,18,6>, -
                               165A                         >
                               168A
                               168A      idb_column_3:
                               168A                column_list -
                               168A                         idb$, 16, 8, 0, <-
                               168A                         <<ADP address>,l_adp,xl>, -
                               168A                         >
                               16AA
                               16AA      ;***********
                               16AA      idb_vector:
        50    08 AB        9A  16AA                movzbl   idb$b_vector(r11), r0              ; Obtain vector information
                 12        13  16AE                beql     90$                                ; Branch if none present
     7E    50    02        78  16B0                ashl     #2, r0, -(sp)                      ; Convert vector information
           52    5E        D0  16B4                movl     sp, r2                             ; Get converted info. addr.
                               16B7                do_column_entry ow                          ; Display information
                 8E        D5  16C0                tstl     (sp)+                              ; Cleanup stack
                           05  16C2      90$:      rsb                                         ; Return to caller
                               16C3
                               16C3      ;
                               16C3      ; PRINT_COLUMNS tables for DDT display
                               16C3      ;
                               16C3
                               16C3      ddt_column_1:
                               16C3                column_list -
                               16C3                         ddt$, 16, 8, 4, <-
```

```
                              16C3  1155                          <<Errlog buf sz>,w_errorbuf,uw>,-
                              16C3  1156                          <<Start I/O>,ddt_address,ddt$l_start>, -
                              16C3  1157                          <<Alt start I/O>,ddt_address,ddt$l_altstart>, -
                              16C3  1158                          <<Cancel I/O>,ddt_address,ddt$l_cancel>, -
                              16C3  1159                          >
                              1713  1160
                              1713  1161 ddt_column_2:
                              1713  1162         column_list -
                              1713  1163                          ddt$, 16, 8, 4, <-
                              1713  1164                          <<Diag buf sz>,w_diagbuf,uw>, -
                              1713  1165                          <<Register dump>,ddt_address,ddt$l_regdump>, -
                              1713  1166                          <<Unit init>,ddt_address,ddt$l_unitinit>, -
                              1713  1167                          <<Unsol int>,ddt_address,ddt$l_unsolint>, -
                              1713  1168                          >
                              1763  1169
                              1763  1170 ddt_column_3:
                              1763  1171         column_list -
                              1763  1172                          ddt$, 16, 8, 0, <-
                              1763  1173                          <<FDT size>,w_fdtsize,uw>, -
                              1763  1174                          <<FDT address>,l_fdt,xl>, -
                              1763  1175                          <<Mnt verify>,ddt_address,ddt$l_mntver>, -
                              1763  1176                          <<Cloned UCB>,ddt_address,ddt$l_cloneducb>, -
                              1763  1177                          >
                              17B3  1178
                              17B3  1179 ;************
                              17B3  1180 ddt_address:
             52     5B    C0  17B3  1181         addl    r11, r2                         ; get datum address
    00000000'EF     62    D1  17B6  1182         cmpl    (r2), ioc$return                ; is this the RSB routine?
                    09    13  17BD  1183         beql    90$                             ; branch if RSB routine
                              17BF  1184         do_column_entry xl, jmp                 ; else, output value
  52   00000B79'EF        9E  17C8  1185 90$:    movab   ddt_return, r2                  ; for RSB routine, display
                              17CF  1186         do_column_entry ac, jmp                 ;  "return"
```

```
                             17D8  1188          .sbttl  show_system_block, show system/path blocks (SB/PB)
                             17D8  1189 ;---
                             17D8  1190 ;
                             17D8  1191 ;        show_system_block
                             17D8  1192 ;
                             17D8  1193 ;        This routine displays the system and path blocks given
                             17D8  1194 ;        the address of the system block.
                             17D8  1195 ;
                             17D8  1196 ;        4(ap) = SVA of the system block of interest
                             17D8  1197 ;---
                             17D8  1198
                             17D8  1199 show_system_block::
                       01FC  17D8  1200          .word   ^m<r2,r3,r4,r5,r6,r7,r8>
   54   00000000'EF      9E  17DA  1201          movab   buffer, r4                             ; get working buffer
                             17E1  1202
                             17E1  1203 ; display system block
                             17E1  1204
                             17E1  1205          ensure  12
                             17F9  1206          getmem  @4(ap), (r4), #sb$k_length             ; copy SB to local mem.
                             180B  1207          retiferr
                04 AC    DD  180F  1208          pushl   4(ap)
                44 A4    9F  1812  1209          pushab  sb$t_nodename(r4)                      ; node name
                             1815  1210          print   1,<!_!_    --- !AC System Block (SB) !XL --->
                             1822  1211          skip    1
                             182B  1212          print_columns -
                             182B  1213                  buffer, 4(ap), -
                             182B  1214                  sb_column_1, sb_column_2
                             1847  1215          skip    1
                             1850  1216
                             1850  1217
                             1850  1218 ; display each path block
                             1850  1219
                             1850  1220          assume  pb$k_length lt 512
        64     0C A4     D0  1850  1221          movl    sb$l_pbfl(r4), pb$l_flink(r4)          ; init PB scan
                             1854  1222
                             1854  1223 pb_loop:
   50   04 AC     0C     C1  1854  1224          addl3   #sb$l_pbfl, 4(ap), r0                  ; is there another PB?
          50      64     D1  1859  1225          cmpl    pb$l_flink(r4), r0
                  03     12  185C  1226          bneq    10$
                0082     31  185E  1227          brw     end_pb                                 ; branch if no PBs left
          58      64     D0  1861  1228 10$:     movl    pb$l_flink(r4), r8                     ; save new PB addr.
                             1864  1229          getmem  (r8), (r4), #pb$k_length               ; copy PB to local mem.
                             1875  1230          retiferr
                             1879  1231          ensure  12
                  58     DD  1891  1232          pushl   r8
                             1893  1233          print   1,<!_!_    --- Path Block (PB) !XL --->
                             18A0  1234          skip    1
          57      5E     D0  18A9  1235          movl    sp, r7                                 ; save stack pointer
                             18AC  1236          alloc   80, r6                                 ; allocate scratch
        7E     44 A4     3C  18BE  1237          movzwl  pb$w_sts(r4), -(sp)                    ; push PB STS
             E73A CF     9F  18C2  1238          pushab  pb_status                              ; push bit conv. data
  00000000'GF     02     FB  18C6  1239          calls   #2, g^translate_bits                   ; translate PB STS
                  56     DD  18CD  1240          pushl   r6                                     ; push result
        7E     44 A4     3C  18CF  1241          movzwl  pb$w_sts(r4), -(sp)                    ; push PB STS
                             18D3  1242          print   2,<!_!_Status:  !XW  !AS>              ; output PB STS
          5E      57     D0  18E0  1243          movl    r7, sp                                 ; restore stack
                             18E3  1244          skip    1
```

```
                    18EC  1245          print_columns -
                    18EC  1246                  buffer, r8, -
                    18EC  1247                  pb_column_1, pb_column_2
                    1907  1248          skip    1
        FF41   31   1910  1249          brw     pb_loop
                    1913  1250
                    1913  1251  end_pb:
               04   1913  1252          ret
```

```
                                1914         .sbttl show_system_block tables & action routines
                                1914
                                1914  ;  The following are all PRINT_COLUMNS action routines for the show
                                1914  ;  system/path block displays.
                                1914  ;
                                1914  ;    Action Routine Inputs:
                                1914  ;
                                1914  ;       R2              value from the COLUMN_LIST entry
                                1914  ;       R5              size of value section for this item
                                1914  ;       R7              address of a descriptor for a scratch string in
                                1914  ;                       which the FAO converted value is to be returned
                                1914  ;       R11             base address of the local UCB copy
                                1914  ;
                                1914  ;    Action Routine Outputs:
                                1914  ;
                                1914  ;       R0              status
                                1914  ;                          lbs ==> use this entry
                                1914  ;                          lbc ==> skip this entry
                                1914  ;       R1 - R5         scratch
                                1914  ;                       all other registers must be preserved
                                1914
                                1914  ;
                                1914  ; FAO control strings, etc. used by the action routines
                                1914  ;
                                1914
                                1914         .save
                            00000DEE         .psect  literals,exe,nowrt
                                0DEE
                                0DEE  sb_fao_6bytes:
                                0DEE         string  <!#* !XW!XL>
                                0E00
                                0E00  sb_fao_ascic:
                                0E00         string  <!#* !#(AC)>
                                0E12
                                0E12  cddb_fao:
                                0E12         string  <!#* !XL>
                                0E21
                                0E21  null_ascic:
                      00000000  0E21         .long   0
                                0E25
                                0E25  maint_ascic:
      5F 54 4E 49 41 4D 00'     0E25         .ascic  /MAINT_/
                         06     0E25
                                0E2C
                                0E2C  cbl_a_ascic:
                   2D 41 00'    0E2C         .ascic  /A-/
                         02     0E2C
                                0E2F
                                0E2F  cbl_b_ascic:
               2D 42 20 00'     0E2F         .ascic  / B-/
                         03     0E2F
                                0E33
                                0E33  ok_ascic:
                   4B 4F 00'    0E33         .ascic  /OK/
                         02     0E33
                                0E36
                                0E36  bad_ascic:
```

```
   44 41 42 00'  0E36            .ascic  /BAD/
             03  0E36
                 0E3A
                 0E3A  crossed_ascic:
64 65 58 20 00'  0E3A            .ascic  / Xed/
             04  0E3A
                 0E3F
             00001914            .restore
                 1914
                 1914  ;
                 1914  ; PRINT_COLUMNS tables for SB display
                 1914  ;
                 1914
                 1914  sb_column_1:
                 1914            column_list -
                 1914                    sb$, 21, 12, 4, < -
                 1914                    <<System ID>,sb_6bytes,sb$b_systemid>, -
                 1914                    <<Max message size>,w_maxmsg,uw>, -
                 1914                    <<Max datagram size>,w_maxdg,uw>, -
                 1914                    <<Local hardware type>,sb_lwchar,sb$t_hwtype,29,4>, -
                 1914                    <<Local hardware vers.>,sb_6bytes,sb$b_hwvers>, -
                 1914                    << >,sb_6bytes,sb$b_hwvers+6>, -
                 1914                    >
                 1984
       00000030  1984  sb$q_swincarn2 = sb$q_swincarn+4
                 1984  sb_column_2:
                 1984            column_list -
                 1984                    sb$, 21, 12, 0, < -
                 1984                    <<Local software type>,sb_lwchar,sb$t_swtype,29,4>, -
                 1984                    <<Local software vers.>,sb_lwchar,sb$t_swvers,29,4>, -
                 1984                    <<Local software incarn.>,q_swincarn,xl,25,8>, -
                 1984                    << >,q_swincarn2,xl,25,8>, -
                 1984                    <<SCS poller timeout>,w_timeout,xw>, -
                 1984                    <<SCS poller enable mask>,b_enbmsk,xb,31,2>, -
                 1984                    >
                 19F4
                 19F4  ;**********
                 19F4  sb_6bytes:
53    5B    52    C1  19F4            addl3   r2, r11, r3                     ; locate storage of interest
      55    0C    C2  19F8            subl    #12, r5                         ; get size of filler field
                 19FB            $fao_s  -
                 19FB                    ctrstr = sb_fao_6bytes, -
                 19FB                    outbuf = (r7), -
                 19FB                    outlen = (r7), -
                 19FB                    p1 = r5, -
                 19FB                    p2 = 4(r3), -
                 19FB                    p3 = (r3)
             05  1A13            rsb
                 1A14
                 1A14  ;**********
                 1A14  sb_lwchar:
53    5B    52    C1  1A14            addl3   r2, r11, r3                     ; locate storage of interest
            7E    D4  1A18            clrl    -(sp)                           ; make scratch ASCIC space
            63    95  1A1A            tstb    (r3)                            ; check for null string
            16    13  1A1C            beql    5$                              ; equal, null string
            04    DD  1A1E            pushl   #4                              ;  of the right size
      52    5E    D0  1A20  10$:      movl    sp, r2                          ; save ASCIC pointer
```

```
              01 A2    63    D0  1A23          movl    (r3), 1(r2)                     ; put text in ASCIC string
                                 1A27          do_column_entry ac                      ; convert the ASCIC
                 5E    08    C0  1A30          addl    #<2*4>, sp                      ; cleanup stack
                             05  1A33          rsb
                                 1A34
                       00    DD  1A34  5$:     pushl   #0
                       E8    11  1A36          brb     10$
                                 1A38
                                 1A38  ;
                                 1A38  ; PRINT_COLUMNS tables for PB display
                                 1A38  ;
                                 1A38
                                 1A38  pb_column_1:
                                 1A38          column_list -
                                 1A38                  pb$, 21, 12, 4, < -
                                 1A38                  <<Remote sta. addr.>,sb_6bytes,pb$b_rstation>, -
                                 1A38                  <<Remote state>,pb_rmtstate,0>, -
                                 1A38                  <<Remote hardware rev.>,l_rport_rev,xl>, -
                                 1A38                  <<Remote func. mask>,l_rport_fcn,xl>, -
                                 1A38                  <<Reseting port>,b_rst_port,xb>, -
                                 1A38                  <<Handshake retry cnt.>,w_retry,uw>, -
                                 1A38                  <<Msg. buf. wait queue>,l_waitqfl,q2>, -
                                 1A38                  >
                                 1AB8
                                 1AB8  pb_column_2:
                                 1AB8          column_list -
                                 1AB8                  pb$, 21, 12, 4, < -
                                 1AB8                  <<Remote port type>,pb_rport_typ,0>, -
                                 1AB8                  <<Number of data paths>,pb_dualpath,0>, -
                                 1AB8                  <<Cables state>,pb_cables,0,18,15>,-
                                 1AB8                  <<Local state>,pb_lclstate,0>, -
                                 1AB8                  <<Port dev. name>,sb_lwchar,pb$t_lport_name,29,4>, -
                                 1AB8                  <<SCS MSGBUF address>,l_scsmsg,xl>, -
                                 1AB8                  <<PDT address>,l_pdt,xl>, -
                                 1AB8                  >
                                 1B38
                                 1B38  ;***********
                                 1B38  pb_rmtstate:
        54   00000E21'EF     9E  1B38          movab   null_ascic, r4                  ; assume rport not in maint.
                                 1B3F          assume  pb$v_maint eq 0                 ;  state
              0A 21 AB       E9  1B3F          blbc    pb$b_rstate(r11), 20$           ; branch if rport not in maint.
        54   00000E25'EF     9E  1B43          movab   maint_ascic, r4                 ; else, set maintenance flag
                 55    64    A2  1B4A          subw    (r4), r5                        ;  and reduce the fill count
            53   E4E7  CF    9E  1B4D  20$:    movab   pb_rstate, r3                   ; get remote state tbl. addr.
  52    21 AB    02    01    EF  1B52          extzv   #pb$v_state, #pb$s_state, -     ; extract remote port state
                                 1B58                  pb$b_rstate(r11), r2            ;  information
           00000000'GF       16  1B58          jsb     g^translate_address             ; convert it to ASCIC pointer
                       1D    13  1B5E          beql    90$                             ; branch if translation failed
                 55    60    82  1B60          subb    (r0), r5                        ; reduce the fill count
                                 1B63          $fao_s  -
                                 1B63                  ctrstr = sb_fao_ascic, -
                                 1B63                  outbuf = (r7), -
                                 1B63                  outlen = (r7), -
                                 1B63                  p1 = r5, -
                                 1B63                  p2 = #2, -
                                 1B63                  p3 = r4, -
                                 1B63                  p4 = r0
                                 05 1B7C             rsb
               52    21 AB       9E 1B7D  90$:       movab   pb$b_rstate(r11), r2             ; if cannot convert remote
                                    1B81             do_column_entry xb, jmp                  ;  status then display value
                                    1B8A
                                    1B8A  ;************
                                    1B8A  pb_rport_typ:
               53  E4CA CF       9E 1B8A             movab   pb_rport_type, r3                ; get port type conversion
                                    1B8F             assume  pb$v_port_typ eq 0
                                    1B8F             assume  pb$s_port_typ eq 31
  52    14 AB    80000000 8F     CB 1B8F             bicl3   #^x80000000, -                   ; get remote port type value
                                    1B98                     pb$l_rport_typ(r11), r2
                 00000000'GF     16 1B98             jsb     g^translate_address              ; translate port type
                          0C     13 1B9E             beql    90$                              ; branch if translation failed
                    52    50     D0 1BA0             movl    r0, r2                           ; setup string for display
                                    1BA3             do_column_entry ac, jmp                  ; display translated string
                                    1BAC
                          52     DD 1BAC  90$:       pushl   r2                               ; else, display just the port
                    52    5E     D0 1BAE             movl    sp, r2                           ;  type value
                                    1BB1             do_column_entry xl
                          8E     D5 1BBA             tstl    (sp)+                            ; cleanup stack
                                 05 1BBC             rsb
                                    1BBD
                                    1BBD  ;***********
                                    1BBD  pb_dualpath:
                                    1BBD             assume  pb$m_dualpath eq <^x80000000>
     52    14 AB    01    1F     EF 1BBD             extzv   #pb$v_dualpath, #1, -            ; get paths flag for remote port
                                    1BC3                     pb$l_rport_typ(r11), r2
               7E    52    01    C1 1BC3             addl3   #1, r2, -(sp)                    ; add one (there's at least one)
                     52    5E    D0 1BC7             movl    sp, r2                           ; get value pointer
                                    1BCA             do_column_entry ul                       ; display value
                          8E     D5 1BD3             tstl    (sp)+                            ; cleanup stack
                                 05 1BD5             rsb
                                    1BD6
                                    1BD6  ;*********
                                    1BD6  pb_cables:
                                    1BD6             assume  pb$v_cur_ps eq 0
                    54    03     D0 1BD6             movl    #3, r4                           ; assume single path port
                 00000E21'EF     9F 1BD9  10$:       pushab  null_ascic
                       F7 54     F5 1BDF             sobgtr  r4, 10$
                                    1BE2
                    55    04     C2 1BE2             subl    #4, r5                           ; adjust fill for path A
                 00000E33'EF     9F 1BE5             pushab  ok_ascic                         ; assume path A is ok
                    09 29 AB     E8 1BEB             blbs    pb$b_p0_sts(r11), 25$            ; branch if path A is ok
           6E    00000E36'EF     9E 1BEF             movab   bad_ascic, (sp)                  ; else, change path A to bad
                          55     D7 1BF6             decl    r5                               ; adjust fill for bad path
                 00000E2C'EF     9F 1BF8  25$:       pushab  cbl_a_ascic                      ; insert "A-"
                                    1BFE
                                    1BFE             assume  pb$m_dualpath eq <^x80000000>
                       14 AB     D5 1BFE             tstl    pb$l_rport_typ(r11)              ; is this a dual pathed port?
                          30     18 1C01             bgeq    40$                              ; branch if not dual pathed
                    55    05     C2 1C03             subl    #5, r5                           ; adjust fill for path B
        0C AE    00000E33'EF     9E 1C06             movab   ok_ascic, 12(sp)                 ; assume path B is ok
                    0A 2A AB     E8 1C0E             blbs    pb$b_p1_sts(r11), 33$            ; branch if path B is ok
        0C AE    00000E36'EF     9E 1C12             movab   bad_ascic, 12(sp)                ; else, change path B to bad
                          55     D7 1C1A             decl    r5                               ; adjust fill for bad path
        08 AE    00000E2F'EF     9E 1C1C  33$:       movab   cbl_b_ascic, 8(sp)               ; add " B-"
                                    1C24             assume  pb$v_cur_cbl eq 0
           0B 28 AB   E8  1C24          blbs    pb$b_cbl_sts(r11), 40$         ; branch if cables not crossed
 10 AE   00000E3A'EF  9E  1C28          movab   crossed_ascic, 16(sp)          ; else, add crossed cables flag
              55  04  C2  1C30          subl    #4, r5                         ;  and adjust fill count
                          1C33
                  05  DD  1C33  40$:    pushl   #5                             ; set number of ASCICs
                  55  DD  1C35          pushl   r5                             ; set fill count
              54  5E  D0  1C37          movl    sp, r4                         ; get parameter list pointer
                          1C3A          $faol_s -
                          1C3A                  ctrstr = sb_fao_ascic, -
                          1C3A                  outbuf = (r7), =
                          1C3A                  outlen = (r7), -
                          1C3A                  prmlst = (r4)
              5E  1C  C0  1C4D          addl    #<7*4>, sp                     ; cleanup stack
                      05  1C50          rsb
                          1C51
                          1C51  ;************
                          1C51  pb_lclstate:
        53   E3BB CF  9E  1C51          movab   pb_state, r3                   ; get port state conversion
                          1C56          assume  pb$v_port_typ eq 0
         52    12 AB  3C  1C56          movzwl  pb$w_state(r11), r2            ; get local port state
       00000000'GF    16  1C5A          jsb     g^translate_address            ; translate port state
                  0C  13  1C60          beql    90$                            ; branch if translation failed
              52  50  D0  1C62          movl    r0, r2                         ; setup string for display
                          1C65          do_column_entry ac, jmp                ; display trans. string
                          1C6E
         52    12 AB  9E  1C6E  90$:    movab   pb$w_state(r11), r2            ; else, display just the port
                          1C72          do_column_entry xw, jmp                ;  state value
```

```
                              1C7B                .sbttl  show_ucb, show unit control block (UCB)
                              1C7B         ;---
                              1C7B         ;
                              1C7B         ;      show_ucb
                              1C7B         ;
                              1C7B         ;      This routine shows the unit control block associated
                              1C7B         ;      with a device.
                              1C7B         ;
                              1C7B         ;
                              1C7B         ;      4(ap) = address of DDB in local storage
                              1C7B         ;      8(ap) = address of UCB in local storage
                              1C7B         ;      12(ap)= actual address of UCB
                              1C7B         ;      16(ap)= address of nodename in local storage
                              1C7B         ;      20(ap)= flags longword
                              1C7B         ;
                              1C7B         ;---
                              1C7B
                              1C7B         show_ucb:
                        0FFC  1C7B                .word   ^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                              1C7D
                              1C7D                ensure  24
              0C AC     DD    1C95                pushl   12(ap)                  ; push virtual address of UCB
                              1C98
        54    08 AC     D0    1C98                movl    8(ap), r4               ; get local address of UCB
        00001160'EF     9F    1C9C                pushab  unknown                 ; assume the device will be unknown
        52    40 A4     9A    1CA2                movzbl  ucb$b_devclass(r4), r2  ; get device class value
      53   E5F6 CF      9E    1CA6                movab   device_class, r3        ; get conversion table
        00000000'GF     16    1CAB                jsb     g^translate_address     ; get address of device type table
                 12     13    1CB1                beql    90$                     ; branch if no class match
        52    41 A4     9A    1CB3                movzbl  ucb$b_devtype(r4), r2   ; get device type value
           53    50     D0    1CB7                movl    r0, r3                  ; get table address picked above
        00000000'GF     16    1CBA                jsb     g^translate_address     ; get device type ASCIC address
                 03     13    1CC0                beql    90$                     ; branch if no device type matches
           6E    50     D0    1CC2                movl    r0, (sp)                ; else replace unknown with devtype
        52    04 AC     7D    1CC5         90$:   movq    4(ap), r2               ; get DDB and UCB addresses
                              1CC9
 5A     00001065'EF     7D    1CC9                movq    one_path, r10           ; assume a single path device which
                              1CD0                                                ; is not a virtual terminal
                              1CD0
    40 A3    42 8F      91    1CD0                cmpb    #dc$_term, -            ; is this a terminal?
                              1CD5                        ucb$b_devclass(r3)
                 49     12    1CD5                bneq    200$                    ; branch if not a terminal
     54    00A0 C3      D0    1CD7                movl    ucb$l_tl_phyucb(r3), r4 ; is this a virtual terminal?
                 3F     13    1CDC                beql    7777$                   ; branch if not a virtual terminal
        0C AC    54     D1    1CDE                cmpl    r4, 12(ap)              ; does virt. term. equal phy. term.?
                 39     13    1CE2                beql    7777$                   ; if yes, then this not a virtual term.
 5A     0000111F'EF     7D    1CE4                movq    virtual_terminal, r10   ; it is a virtual terminal
                              1CEB                getmem  ucb$w_unit(r4)          ; get physical terminal's unit number
           7E    51     3C    1CF5                movzwl  r1, -(sp)               ; push than unit number
 55     000000B5'EF     9E    1CF8                movab   ddb_2p, r5              ; get work space for phy. DDB copy
                              1CFF                getmem  ucb$l_ddb(r4)           ; get address of DDB for phy. UCB
                              1D09                getmem  (r1), -(r5), -          ; get local copy of physical DDB
                              1D09                        #ddb$k_length
              14 A5     9F    1D1A                pushab  ddb$t_name(r5)          ; push address of phy. device name
               00A4     31    1D1D         7777$: brw     setup_primary           ; go setup virtual terminal name
                              1D20                                                ; (this is also a branch assist)
                              1D20
```

```
             3C A3    10   D3  1D20  1561 200$:    bitl    #dev$m_2p, -               ; dual path device?
                               1D24  1562                  ucb$l_devchar2(r3)
                      F7   13  1D24  1563          beql    7777$                      ; branch if not dual path
                               1D26  1564
         5A   0000109B'EF  7D  1D26  1565          movq    this_primary, r10          ; assume this path is primary
         59   00000060'EF  9E  1D2D  1566          movab   nodnam_2p, r9              ; get node name workarea address
            54    00A8 C3  D0  1D34  1567          movl    ucb$l_dp_altucb(r3), r4    ; is there a local path?
                      1B   12  1D39  1568          bneq    local_2p_device            ; branch if local path
                               1D3B  1569
                               1D3B  1570                                             ; both paths through the class driver
             7E    54 A3   3C  1D3B  1571          movzwl  ucb$w_unit(r3), -(sp)      ; push secondary unit number
           55     00A0 C3  D0  1D3F  1572          movl    ucb$l_dp_ddb(r3), r5       ; get secondary DDB address
          33 14 AC    01   E1  1D44  1573          bbc     #flag_v_alt_path, -        ; if scanning primary DDB chain,
                               1D49  1574                  20(ap), process_2p_ddb     ;  go join common code
             55    28 A3   D0  1D49  1575          movl    ucb$l_ddb(r3), r5          ; else, other DDB is primary DDB
         5A   000010DD'EF  7D  1D4D  1576          movq    this_secondary, r10        ;  and this is the secondary path
                      26   11  1D54  1577          brb     process_2p_ddb             ; go to common other path code
                               1D56  1578
                               1D56  1579 local_2p_device:                            ; only one path through the class driver
                               1D56  1580          getmem  ucb$w_unit(r4)             ; get other path unit number
                7E    51   3C  1D60  1581          movzwl  r1, -(sp)                  ; push other path unit number
                               1D63  1582          getmem  ucb$l_ddb(r4)              ; get other path ddb address
                55    51   D0  1D6D  1583          movl    r1, r5                     ; save ddb address in right place
          07 3C A3    03   E1  1D70  1584          bbc     #dev$v_cdp, -              ; branch if the path whose UCB is in
                               1D75  1585                  ucb$l_devchar2(r3), -      ; r3 is the primary path
                               1D75  1586                  process_2p_ddb             ; else indicate that first name is
         5A   000010DD'EF  7D  1D75  1587          movq    this_secondary, r10        ; the secondary path
                               1D7C  1588
                               1D7C  1589 process_2p_ddb:
         54   000000B5'EF  9E  1D7C  1590          movab   ddb_2p, r4                 ; get workarea address for 2p DDB
                               1D83  1591          getmem  (r5), (r4), -              ; pickup secondary DDB
                               1D83  1592                  #ddb$k_length
                   14 A4   9F  1D94  1593          pushab  ddb$t_name(r4)             ; push address of secondary device name
         59   00000060'EF  9E  1D97  1594          movab   nodnam_2p, r9              ; get workarea address fo 2p node name
 50    34 A4  00000044 8F  C1  1D9E  1595          addl3   #sb$t_nodename, -          ; locate secondary node name
                               1DA7  1596                  ddb$l_sb(r4), r0
                               1DA7  1597          getmem  (r0), (r9), -              ; pickup secondary node name
                               1DA7  1598                  #sb$s_nodename
                51    51   9A  1DB4  1599          movzbl  r1, r1                     ; convert byte count to long word
                      0B   13  1DB7  1600          beql    setup_primary              ; don't add "$" to null node name
                      51   D6  1DB9  1601          incl    r1                         ; add one for "$"
                69    51   90  1DBB  1602          movb    r1, (r9)                   ; store count in ASCIC string
              6941    24   90  1DBE  1603          movb    #^a/$/, (r9)[r1]           ; store "$" in string
                      59   DD  1DC2  1604          pushl   r9                         ; push node name pointer
                               1DC4  1605
                               1DC4  1606 setup_primary:
                   54 A3   DD  1DC4  1607          pushl   ucb$w_unit(r3)             ; unit number
                   14 A2   DF  1DC7  1608          pushal  ddb$t_name(r2)             ; generic controller name
                   10 AC   DD  1DCA  1609          pushl   16(ap)                     ; address of nodename
                               1DCD  1610          printd  r10, (r11)                 ; print device name and UCB
                               1DD8  1611          skip    1
                5B    5E   D0  1DE1  1612          movl    sp, r11                    ; save pre-allocation stack pointer
                               1DE4  1613          alloc   80, r4                     ; allocate an output buffer
                   64 A3   DD  1DF6  1614          pushl   ucb$l_sts(r3)              ; push device status value
                 E2BB CF   9F  1DF9  1615          pushab  unit_status                ; bit definition table
         00000000'EF  02   FB  1DFD  1616          calls   #2,translate_bits          ; translate bits into string
                      54   DD  1E04  1617          pushl   r4                         ; result string
                   64 A3    DD  1E06   1618            pushl   ucb$l_sts(r3)           ; push device status value
                                1E09   1619            print   2,<Device status:   !XL !AS>
            64     50 8F    9A  1E16   1620            movzbl  #80, (r4)               ; refresh output buffer descriptor
                   38 A3    DD  1E1A   1621            pushl   ucb$l_devchar(r3)       ; push device characteristics one
                 E33F CF    9F  1E1D   1622            pushab  device_char             ; setup bit definition table
       00000000'EF    02    FB  1E21   1623            calls   #2, translate_bits      ; translate bits into string
                      54    DD  1E28   1624            pushl   r4                      ; push result string
                   38 A3    DD  1E2A   1625            pushl   ucb$l_devchar(r3)       ; push device characteristics one
                                1E2D   1626            print   2,<Characteristics: !XL !AS>
            64     50 8F    9A  1E3A   1627            movzbl  #80, (r4)               ; refresh output buffer descriptor
                   3C A3    DD  1E3E   1628            pushl   ucb$l_devchar2(r3)      ; push device characteristics two
                 E403 CF    9F  1E41   1629            pushab  device_char_2           ; setup bit definition table
       00000000'EF    02    FB  1E45   1630            calls   #2, translate_bits      ; translate bits into string
                      54    DD  1E4C   1631            pushl   r4                      ; push result string
                   3C A3    DD  1E4E   1632            pushl   ucb$l_devchar2(r3)      ; push device characteristics two
                                1E51   1633            print   2,<                 !XL !AS>
              5E   5B       D0  1E5E   1634            movl    r11, sp                 ; restore stack pointer
                                1E61   1635            skip    1
                                1E6A   1636
                                1E6A   1637 define_ucb_symbols:
                                1E6A   1638            .enable lsb
                                1E6A   1639            make_symbol UCB, 12(ap)
                                1E80   1640            make_symbol SB, ddb$l_sb(r2)
                                1E96   1641            make_symbol ORB, ucb$l_orb(r3)
                                1EAC   1642            make_symbol DDB, ucb$l_ddb(r3)
                                1EC2   1643            make_symbol DDT, ucb$l_ddt(r3)
                                1ED9   1644            make_symbol CRB, ucb$l_crb(r3)
                   60 A3    D5  1EEF   1645            tstl    ucb$l_amb(r3)
                      16    13  1EF2   1646            beql    10$
                                1EF4   1647            make_symbol AMB, ucb$l_amb(r3)
         16 64 A3    08     E1  1F0A   1648 10$:       bbc     #ucb$v_bsy, ucb$l_sts(r3), 20$
                                1F0F   1649            make_symbol IRP, ucb$l_irp(r3)
                                1F25   1650 20$:
                                1F25   1651            .disable lsb
                                1F25   1652
                                1F25   1653 do_ucb_columns:
    0000057C'EF    04 AC    D0  1F25   1654            movl    4(ap), ucb_ddb          ; setup local DDB copy address
                                1F2D   1655            print_columns -
                                1F2D   1656                    @8(ap), 12(ap), -
                                1F2D   1657                    ucb_column_1, ucb_column_2, ucb_column_3
            7E     08 AC    7D  1F4C   1658            movq    8(ap),-(sp)             ; push local,real address of UCB
         25 3C A3    05     E1  1F50   1659            bbc     #dev$v_mscp, ucb$l_devchar2(r3), 30$      ; check to see if mscp ser
       00000575'EF    00    B0  1F55   1660            movw    #0,flag_2nd_cddb        ; initialize flag to zero for primary
            56   00BC C3    D0  1F5C   1661            movl    ucb$l_cddb(r3),r6       ; pass the address of the cddb by reg. 6
       0000313C'EF    63    FA  1F61   1662            callg   (r3),show_cddb          ; Display class driver data block
          00000575'EF       B6  1F68   1663            incw    flag_2nd_cddb           ; set to 1 to indicate secondary
            56   00C0 C3    D0  1F6E   1664            movl    ucb$l_2p_cddb(r3),r6    ; pass the address of the secondary cddb
       0000313C'EF    63    FA  1F73   1665            callg   (r3),show_cddb          ; Display class driver data block
       000024C9'EF    02    FB  1F7A   1666 30$:       calls   #2,show_ioq             ; Display I/O request queue
       00002AB1'EF    63    FA  1F81   1667            callg   (r3),show_vcb           ; Display volume control block
                            04  1F88   1668            ret
```

```
                                    1F89   1670            .sbttl get_ucb, copy UCB to local storage
                                    1F89   1671
                                    1F89   1672 ; This routine knows how to load enough of the UCB into local memory for
                                    1F89   1673 ; the operations performed above, but how to avoid trying to load more
                                    1F89   1674 ; UCB than there really is.
                                    1F89   1675 ;
                                    1F89   1676 ; Inputs:
                                    1F89   1677 ;
                                    1F89   1678 ;          r2      real UCB address
                                    1F89   1679 ;          r7      address of the place to copy it to
                                    1F89   1680 ;
                                    1F89   1681 ; Outputs:
                                    1F89   1682 ;
                                    1F89   1683 ;          r0      status of the copy operation
                                    1F89   1684 ;          r1      first longword of copied UCB
                                    1F89   1685
                                    1F89   1686
                                    1F89   1687 get_ucb:
                              3C   BB   1F89   1688            pushr   #^m<r2,r3,r4,r5>          ; save registers
 67   00CC 8F   00    6E      00   2C   1F8B   1689            movc5   #0,(sp),#0,#ucb_size,(r7)   ; zero out the local ucb
                              3C   BA   1F93   1690            popr    #^m<r2,r3,r4,r5>          ; restore registers
                                        1F95   1691            trymem  ucb$w_size(r2)            ; get size of this UCB
                         1E   50   E9   1F9F   1692            blbc    r0, 90$                   ; exit now, if error occured
                         51   51   3C   1FA2   1693            movzwl  r1, r1                    ; extend size to a longword
              000000CC 8F     51   D1   1FA5   1694            cmpl    r1, #ucb_size             ; is UCB bigger than the local space?
                              05   15   1FAC   1695            bleq    10$                       ; branch if not bigger
           51    00CC 8F           3C   1FAE   1696            movzwl  #ucb_size, r1             ; else minimize the size
                                        1FB3   1697 10$:       trymem  (r2), (r7), r1            ; copy UCB to local storage
                                   05   1FC0   1698 90$:       rsb                               ; return to caller
```

```
                                 1FC1   1700          .sbttl show_ucb tables & action routines
                                 1FC1   1701
                                 1FC1   1702          .save
                             00001065   1703          .psect  literals,exe,nowrt
                                 1065   1704
                                 1065   1705  ;
                                 1065   1706  ; FAO control strings for locally generated UCB displays
                                 1065   1707  ;
                                 1065   1708
                                 1065   1709  one_path:
           0000106D'00000005'    1065   1710          .address 5, 10$
                                 106D   1711  10$:    string  ^\!40<!AC!AC!UW!>!17AC UCB address:  !XL\
                                 109B   1712
                                 109B   1713  this_primary:
           000010A3'00000008'    109B   1714          .address 8, 10$
                                 10A3   1715  10$:    string  ^\!40<!AC!AC!UW (!AC!AC!UW)!>!17AC UCB address:  !XL\
                                 10DD   1716
                                 10DD   1717  this_secondary:
           000010E5'00000008'    10DD   1718          .address 8, 10$
                                 10E5   1719  10$:    string  ^\!40<(!AC!AC!UW) !AC!AC!UW!>!17AC UCB address:  !XL\
                                 111F   1720
                                 111F   1721  virtual_terminal:
           00001127'00000007'    111F   1722          .address 7, 10$
                                 1127   1723  10$:    string  ^\!40<!AC!AC!UW ==> !AC!UW!>!17AC UCB address:  !XL\
                                 1160   1724
                                 1160   1725  unknown:
   6E 77 6F 6E 6B 6E 55 00'      1160   1726          .ascic  /Unknown/
                         07      1160
                                 1168   1727
                                 1168   1728  ;
                                 1168   1729  ; FAO control strings used by the action routines
                                 1168   1730  ;
                                 1168   1731
                                 1168   1732  ucb_uic_cstr1:
                                 1168   1733          string  <[!6OW,!6OW]>
                                 117B   1734
                                 117B   1735  ucb_two_bytes:
                                 117B   1736          string  <!5XB/!2XB>
                                 118C   1737
                                 118C   1738  ucb_retry_fao:
                                 118C   1739          string  <!#UB/!UB>
                                 119C   1740
                                 119C   1741  ucb_test_retry_fao:
                                 119C   1742          string  <!UB>
                                 11A7   1743
                             00001FC1   1744          .restore
                                 1FC1   1745
                                 1FC1   1746  ;
                                 1FC1   1747  ;  PRINT_COLUMNS tables for UCB display
                                 1FC1   1748  ;
                                 1FC1   1749
                                 1FC1   1750  ucb_column_1:
                                 1FC1   1751          column_list -                           ; column 1 -- allocation
                                 1FC1   1752                  ucb$, 17, 8, 3, < -             ;   and other device status
                                 1FC1   1753                  <<Owner UIC>,orb_owner,0,10,15>, -      ; Owner UIC
                                 1FC1   1754                  <<      PID>,l_pid,xl>, -               ; Owner PID
                                 1FC1   1755                  <<Alloc. lock ID>,ucb_lockid,0>, -      ; Allocation lock ID
```

```
                    1FC1                       -                                   ; Allocation class
                    1FC1                       <<Alloc. class>,ucb_alloclass,ucb_ddb>, -
                    1FC1                       <<Class/Type>,ucb_clstyp,0>, -      ; Device class/type
                    1FC1                       <<Def. buf. size>,w_devbufsiz,uw>, - ; default buffer size
                    1FC1                       <<DEVDEPEND>,l_devdepend,xl>, -     ; Device dependent first
                    1FC1                       <<DEVDEPND2>,l_devdepnd2,xl>, -     ;   "        "      sec.
                    1FC1                       <<FIPL/DIPL>,ucb_ipls,0>, -         ; Fork / Device IPL
                    1FC1                       <<Charge PID>,ucb_cpid,0>, -        ; UCB size charge PID
                    1FC1                       >                               ; *** end column 1
                    2071
                    2071               .save
                0000057C               .psect  sdadata,noexe,wrt
                    057C       ucb_ddb:
      00000000      057C               .long   0
                00002071               .restore
                    2071
                    2071       ucb_column_2:
                    2071               column_list -                       ; column 2 -- device activity
                    2071                       ucb$, 18, 8, 3, < -         ;    data
                    2071                       <<Operation count>,l_opcnt,ul>, - ; operations completed
                    2071                       <<Error count>,w_errcnt,uw>, - ; errors recorded count
                    2071                       <<Reference count>,w_refc,uw>, -; reference count
                    2071                       <<Online count>,ucb_onlcnt,0>, -; online count
                    2071                       <<Retry cnt/max>,ucb_retry,0>, -; error retry count/maximum
                    2071                       <<BOFF>,w_boff,xw>, -        ; byte offset
                    2071                       <<Byte count>,w_bcnt,xw>, -  ; byte count
                    2071                       <<SVAPTE>,l_svapte,xl>, -    ; system virtual addr. PTE
                    2071                       <<SVPN>,ucb_svpn,0> -        ; system virtual page number
                    2071                       <<DEVSTS>,w_devsts,xw>, -    ; Device dependent status
                    2071                       <<Master CSID>,ucb_mcsid,0>, - ; Master node's CSID
                    2071                       <<Int. due time>,ucb_duetim,0>, - ; Interrupt due time
                    2071                       <<RWAITCNT>,ucb_rwaitcnt,0> - ; Reasons to wait count
                    2071                       >                           ; *** end column 2
                    2151
                    2151       ucb_column_3:
                    2151               column_list -                       ; column 3 -- pointer addresses
                    2151                       ucb$, 15, 8, 0, < -
                    2151                       <<ORB address>,l_orb,xl>, -  ; Object's rights block
                    2151                       <<DDB address>,l_ddb,xl>, -  ; Device data block
                    2151                       <<DDT address>,l_ddt,xl>, -  ; Driver dispatch table
                    2151                       <<VCB address>,ucb_vcb,0>, - ; Volume control block
                    2151                       <<CRB address>,l_crb,xl>, -  ; Channel request block
                    2151                       <<LNM address>,ucb_lnm,0>, - ; MBX LNM pointer
                    2151                       <<AMB address>,l_amb,xl_neq>, - ; Associated mailbox
                    2151                       <<PDT address>,ucb_pdt,0>, - ; Port descriptor table
                    2151                       <<CDDB address>,ucb_cddb,0>,- ; Class driver data block
                    2151                       <<2P_CDDB addr.>,ucb_2pcddb,0>,-; Alternate CDDB
                    2151                       <<2P_DDB address>,ucb_2pddb,0>, -; Secondary path DDB
                    2151                       <<2P_UCB address>,ucb_altucb,0>,-; Alternate UCB
                    2151                       -                           ; All of the following appear
                    2151                       -                           ;  only when the UCB is busy
                    2151                       <<IRP address>,ucb_bsy,ucb$l_irp>, -    ; I/O request packet
                    2151                       <<Fork PC>,ucb_bsy,ucb$l_fpc>, -        ; Fork PC
                    2151                       <<Fork R3>,ucb_bsy,ucb$l_fr3>, -        ; Fork R3
                    2151                       <<Fork R4>,ucb_bsy,ucb$l_fr4>, -        ; Fork R4
                    2151                       <<I/O wait queue>,l_ioqfl,q2>, -        ; Pending I/O queue
                    2151                       >                           ; *** end column 3
```

```
                        2271
                        2271    ;  The following are all PRINT_COLUMNS action routines for the UCB
                        2271    ;  display.
                        2271    ;
                        2271    ;    Action Routine Inputs:
                        2271    ;
                        2271    ;       R2                 value from the COLUMN_LIST entry
                        2271    ;       R5                 size of value section for this item
                        2271    ;       R7                 address of a descriptor for a scratch string in
                        2271    ;                          which the FAO converted value is to be returned
                        2271    ;       R11                base address of the local UCB copy
                        2271    ;
                        2271    ;    Action Routine Outputs:
                        2271    ;
                        2271    ;       R0                 status
                        2271    ;                             lbs ==> use this entry
                        2271    ;                             lbc ==> skip this entry
                        2271    ;       R1 - R5            scratch
                        2271    ;                          all other registers must be preserved
                        2271
                        2271    ;**************
                        2271    ucb_alloclass:  ; if appropriate, return allocation class
  5F 38 AB   0E    E1   2271            bbc     #dev$v_fod, -                   ; branch if not a file oriented
                        2276                    ucb$l_devchar(r11), ucb_act_nop ;  device
   52   62   3C    C1   2276            addl3   #ddb$l_allocls, (r2), r2        ; get allocation class address
                        227A    ucb_act_ub:                                     ; display allocation class
                        227A            do_column_entry ub, jmp
                        2283
                        2283    ;***********
                        2283    ucb_altucb:
  4D 3C AB   04    E1   2283            bbc     #dev$v_2p, ucb$l_devchar2(r11), - ; branch if device is not
                        2288                    ucb_act_nop                     ;   dual pathed
   52   00A8 CB    DE   2288            moval   ucb$l_dp_altucb(r11), r2        ; alternate UCB address
             62    D5   228D            tstl    (r2)                            ; is there something there?
             44    13   228F            beql    ucb_act_nop                     ; branch if nothing there
                        2291            make_symbol -                           ; else,
                        2291                    2P_UCB, (r2)                    ;  make a symbol and
           0087    31   22A6            brw     ucb_act_xl                      ;  display it
                        22A9
                        22A9    ;%%%%%%%
                        22A9    ucb_bsy:
  27 64 AB   08    E1   22A9            bbc     #ucb$v_bsy, ucb$l_sts(r11), -   ; exit doing nothing if the
                        22AE                    ucb_act_nop                     ;  UCB is not busy
        52   5B    C0   22AE            addl    r11, r2                         ; else locate cell to return
                        22B1    ucb_act_xl_neq:
                        22B1            do_column_entry xl_neq, jmp             ; display that entry
                        22BB
                        22BB    ;***********
                        22BB    ucb_clstyp:     ; return device class / type
     52   40 AB    9A   22BB            movzbl  ucb$b_devclass(r11), r2         ; return device class
     53   41 AB    9A   22BF            movzbl  ucb$b_devtype(r11), r3          ;  and device type
             26    11   22C3            brb     ucb_ret_2xbytes                 ; go join common code
                        22C5
                        22C5    ;*********
                        22C5    ucb_cpid:       ; if appropriate, return PID charged for UCB creation
38 AB  00102000 8F D3   22C5            bitl    #<dev$m_mbx ! dev$m_net>, -     ; is this a mailbox or a
                        22CD                    ucb$l_devchar(r11)              ;  network device
```

```
            06   13 22CD                 beql    ucb_act_nop                     ; if not, assume no PID charged
   52    20 AB   DE 22CF                 moval   ucb$l_cpid(r11), r2             ; else, return charged PID
            5B   11 22D3                 brb     ucb_act_xl                      ; using common code
                    22D5
                    22D5     ucb_act_nop:
            50   D4 22D5                 clrl    r0                              ; make this call a nop
                 05 22D7                 rsb                                     ; return
                    22D8
                    22D8     ;**********
                    22D8     ucb_duetim:         ; if appropriate, return interrupt due time
F8 64 AB    00   E1 22D8                 bbc     #ucb$v_tim, -                   ; branch if time-out not
                    22DD                         ucb$l_sts(r11), ucb_act_nop     ;  expected
   52    6C AB   DE 22DD                 moval   ucb$l_duetim(r11), r2           ; else return due time
            4D   11 22E1                 brb     ucb_act_xl                      ; join common code
                    22E3
                    22E3     ;********
                    22E3     ucb_ipls:           ; return fork / device IPL
   52    0B AB   9A 22E3                 movzbl  ucb$b_fipl(r11), r2             ; return fork IPL
   53    5E AB   9A 22E7                 movzbl  ucb$b_dipl(r11), r3             ;  and device IPL
                    22EB     ucb_ret_2xbytes:
                    22EB                 $fao_s  -                               ; two values as requested
                    22EB                         ctrstr = ucb_two_bytes,  -
                    22EB                         outbuf = (r7), -
                    22EB                         outlen = (r7), -
                    22EB                         p1 = r2, -
                    22EB                         p2 = r3
                 05 2300                 rsb                                     ; return
                    2301
                    2301     ;*******
                    2301     ucb_lnm:
40 AB    A0 8F   91 2301                 cmpb    #dc$_mailbox, -                 ; is this a mailbox?
                    2306                         ucb$b_devclass(r11)
            CD   12 2306                 bneq    ucb_act_nop                     ; branch if not a mailbox
   52    74 AB   DE 2308                 moval   ucb$l_logadr(r11), r2           ; get logical name pointer
            62   D5 230C                 tstl    (r2)                            ; is something there?
            C5   13 230E                 beql    ucb_act_nop                     ; branch if nothing there
                    2310                 make_symbol -                           ; else,
                    2310                         LNM, (r2)                       ;  make a symbol and
            09   11 2325                 brb     ucb_act_xl                      ;  display it
                    2327
                    2327     ;**********
                    2327     ucb_lockid:         ; if sensible, return allocation lock id
A9 3C AB    00   E1 2327                 bbc     #dev$v_clu, -                   ; branch if not a cluster
                    232C                         ucb$l_devchar2(r11), ucb_act_nop ;  accessible device
   52    20 AB   DE 232C                 moval   ucb$l_lockid(r11), r2           ; else return lock id
                    2330     ucb_act_xl:
                    2330                 do_column_entry xl, jmp
                    2339
                    2339     ;*********
                    2339     ucb_mcsid:
40 AB    A1 8F   91 2339                 cmpb    #dc$_journal, -                 ; is this a journal device?
                    233E                         ucb$b_devclass(r11)
            95   12 233E                 bneq    ucb_act_nop                     ; branch if not a journal dev.
  52   0084 CB   DE 2340                 moval   ucb$l_jnl_mcsid(r11), r2        ; else, return master CSID
            E9   11 2345                 brb     ucb_act_xl                      ;  using common code
                    2347
                    2347     ;**********
```

```
                           2347  ucb_onlcnt:
   40 AB    01     91      2347            cmpb    #dc$_disk, ucb$b_devclass(r11)  ; is this a disk device?
            70     12      234B            bneq    ucb_act_nop_a                   ; branch if not a disk
52    00AE  CB     9E      234D            movab   ucb$b_onlcnt(r11), r2           ; else get online count addr.
          FF25     31      2352            brw     ucb_act_ub                      ;  and display it
                           2355
                           2355  ;**********
                           2355  orb_owner:        ; attempt to format owner UIC
            7E     D4      2355            clrl    -(sp)                           ; storage for the UIC from ORB
       52   5E     D0      2357            movl    sp,r2                           ; save address for later
51     1C   AB     D0      235A            movl    ucb$l_orb(r11),r1               ; get real ORB address
            0F     13      235E            beql    10$                             ; display [0,0] if no ORB
                           2360            getmem  orb$l_owner(r1)                 ; get the owner UIC
       03   50     E9      2369            blbc    r0,10$                          ; display [0,0] if unaccessable
       62   51     D0      236C            movl    r1,(r2)                         ; save for $FAO below
                           236F            ASSUME  ORB$L_OWNER EQ 0
                           236F  10$:      $fao_s  -                               ; convert UIC to octal
                           236F                    ctrstr = ucb_uic_cstr1, -
                           236F                    outbuf = (r7), -
                           236F                    outlen = (r7), -
                           236F                    p1 = orb$w_uicgroup(r2), -
                           236F                    p2 = orb$w_uicmember(r2)
            8E     D5      2385            tstl    (sp)+                           ; clean the stack
            05             2387            rsb                                     ; return
                           2388
                           2388  ;*******
                           2388  ucb_pdt:
53    0084  CB     D0      2388            movl    ucb$l_pdt(r11), r3              ; get possible PDT address
            2E     13      238D            beql    ucb_act_nop_a                   ; branch if none
                           238F            getmem  ucb$b_type(r3)                  ; get type and sub-type of PDT
51    0560  8F     B1      2399            cmpw    #<dyn$c_scs_pdt@8 -             ; is thing pointed to really
                           239E                     + dyn$c_scs>, r1               ;  a PDT?
            1D     12      239E            bneq    ucb_act_nop_a                   ; branch if not really a PDT
52    0084  CB     DE      23A0            moval   ucb$l_pdt(r11), r2              ; get address of PDT pointer
                           23A5            make_symbol -
                           23A5                    PDT, (r2)                       ;  make a symbol and
          FF73     31      23BA            brw     ucb_act_xl                      ;  display it
                           23BD
                           23BD
                           23BD  ucb_act_nop_a:
            50     D4      23BD            clrl    r0
            05             23BF            rsb
                           23C0
                           23C0  ;*********
                           23C0  ucb_cddb:
   3C AB    05     E1      23C0            bbc     #dev$v_mscp,ucb$l_devchar2(r11),-
            F8             23C4                    ucb_act_nop_a                   ; branch if device is not mscp serve
52    00BC  CB     DE      23C5            moval   ucb$l_cddb(r11),r2              ; get address of CDDB pointer
                           23CA            make_symbol -
                           23CA                    CDDB, (r2)                      ; make a symbol and
          FF4E     31      23DF            brw     ucb_act_xl                      ; display it
                           23E2
                           23E2  ;*********
                           23E2  ucb_2pcddb:
   3C AB    05     E1      23E2            bbc     #dev$v_mscp,ucb$l_devchar2(r11),-
            D6             23E6                    ucb_act_nop_a                   ; branch if device is not mscp serve
52    00C0  CB     DE      23E7            moval   ucb$l_2p_cddb(r11),r2           ; alternate CDDB address
```

```
               62  D5 23EC  1984           tstl    (r2)                            ; is there a secondary cddb
               CD  13 23EE  1985           beql    ucb_act_nop_a                   ; branch if not
                      23F0  1986           make_symbol -
                      23F0  1987                   2P_CDDB, (r2)                   ; make a symbol and
             FF28  31 2405  1988           brw     ucb_act_xl                      ; display it
                      2408  1989
                      2408  1990   ;*********
                      2408  1991   ucb_retry:
          0081 CB  95 2408  1992           tstb    ucb$b_ertmax(r11)               ; is there a retry max?
               AF  13 240C  1993           beql    ucb_act_nop_a                   ; quit now, if no retry max
               7E  D4 240E  1994           clrl    -(sp)                           ; make a little room on stack
          52   5E  D0 2410  1995           movl    sp, r2                          ; save its address
                      2413  1996           $fao_s  -
                      2413  1997                   ctrstr = ucb_test_retry_fao, -  ; determine size of
                      2413  1998                   outbuf = (r7), -                ;  retry max
                      2413  1999                   outlen = (r2), -
                      2413  2000                   p1 = ucb$b_ertmax(r11)
               6E  D6 2428  2001           incl    (sp)                            ; add one to retry max size
          55   8E  C2 242A  2002           subl    (sp)+, r5                       ; reduce retry cnt. size by that
                      242D  2003           $fao_s  -
                      242D  2004                   ctrstr = ucb_retry_fao, -       ; now produce the whole value
                      242D  2005                   outbuf = (r7), -
                      242D  2006                   outlen = (r7), -
                      242D  2007                   p1 = r5, -
                      242D  2008                   p2 = ucb$b_ertcnt(r11), -
                      242D  2009                   p3 = ucb$b_ertmax(r11)
                   05 2448  2010           rsb                                     ; then return
                      2449  2011
                      2449  2012   ;************
                      2449  2013   ucb_rwaitcnt:
       3C AB   05  E1 2449  2014           bbc     #dev$v_mscp,ucb$l_devchar2(r11),-
               78     244D  2015                   ucb_act_nop_b                   ; branch if device is not mscp serve
          52   56 AB  DE 244E  2016        moval   ucb$w_rwaitcnt(r11),r2          ; get address of wait count
                      2452  2017           make_symbol -
                      2452  2018                   RWAITCNT,(r2)                   ; make a symbol and
                      2467  2019   ucb_act_xw:
                      2467  2020           do_column_entry xw,jmp
                      2470  2021
                      2470  2022   ;********
                      2470  2023   ucb_svpn:
    40 AB   A0 8F  91 2470  2024           cmpb    #dc$_mailbox, -                 ; is this a mailbox? (they
                      2475  2025                   ucb$b_devclass(r11)             ;  don't have SVPN's)
               4F  13 2475  2026           beql    ucb_act_nop_b                   ; branch if mailbox
          52   74 AB  DE 2477  2027        moval   ucb$l_svpn(r11), r2             ; get SVPN address
             FE33  31 247B  2028           brw     ucb_act_xl_neq                  ;  display it if non-zero
                      247E  2029
                      247E  2030   ;*******
                      247E  2031   ucb_vcb:
 38 AB   00280000 8F  D3 247E  2032        bitl    #<dev$m_mnt ! dev$m_dmt>, -     ; is the device mounted?
                      2486  2033                   ucb$l_devchar(r11)
               3E  13 2486  2034           beql    ucb_act_nop_b                   ; branch if not mounted
          52   34 AB  DE 2488  2035        moval   ucb$l_vcb(r11), r2
                      248C  2036           make_symbol -                           ; else,
                      248C  2037                   VCB, (r2)                       ;  make a symbol and
             FE8C  31 24A1  2038           brw     ucb_act_xl                      ;  and display it
                      24A4  2039
                      24A4  2040   ;*********
```

```
                          24A4   2041 ucb_2pddb:
 1D 3C AB    04    E1     24A4   2042            bbc     #dev$v_2p, ucb$l_devchar2(r11), - ; branch if device is not
                          24A9   2043                    ucb_act_nop_b                      ;  dual pathed
  52    00A0 CB    DE     24A9   2044            moval   ucb$l_dp_ddb(r11), r2              ; secondary DDB address
                          24AE   2045            make_symbol -
                          24AE   2046                    2P_DDB, (r2)                       ; make a symbol and
           FDEB    31     24C3   2047            brw     ucb_act_xl_neq                     ;  display it
                          24C6   2048
                          24C6   2049 ucb_act_nop_b:
             50    D4     24C6   2050            clrl    r0
                   05     24C8   2051            rsb
                          24C9   2052
```

```
                                  24C9          .sbttl  show_ioq, Display I/O queue for device
                                  24C9   ;---
                                  24C9   ;
                                  24C9   ;       show_ioq
                                  24C9   ;
                                  24C9   ;       Display the IRPs and/or CDRP's (if mscp served) in the I/O queues
                                  24C9   ;       associated with a specified device.
                                  24C9   ;
                                  24C9   ; Inputs:
                                  24C9   ;
                                  24C9   ;       4(ap) = Address of UCB in local storage
                                  24C9   ;       8(ap) = Actual address of UCB
                                  24C9   ;
                                  24C9   ;---
                                  24C9          .enabl  lsb
                                  24C9
                                  24C9   show_ioq:
                           01FC   24C9          .word   ^m<r2,r3,r4,r5,r6,r7,r8>
                                  24CB
           52   04 AC      D0     24CB          movl    4(ap),r2                        ; address of UCB
      34 3C A2   05        E1     24CF          bbc     #dev$v_mscp,ucb$l_devchar2(r2),5$
                                  24D4                                                  ; only 1 queue if not mscp served
     57    00000471'EF     9E     24D4          movab   cddb,r7                         ; address of Class Driver Data Block
                                  24DB          getmem  @ucb$l_cddb(r2),(r7),#cddb$c_length    ; read CDDB
           4D 50           E9     24EE          blbc    r0,8$                           ; branch if cannot read entire CDDB
     54   00BC C2   00     C1     24F1          addl3   #cddb$l_cdrpqfl,ucb$l_cddb(r2),r4       ; Get real address of cdrp q
              54   67      D1     24F7          cmpl    cddb$l_cdrpqfl(r7),r4    ; Empty CDRP queue?
                   45      12     24FA          bneq    10$                      ; branch if not empty
     54   00BC C2   3C     C1     24FC   4$:    addl3   #cddb$l_rstrtqfl,ucb$l_cddb(r2),r4  ; Get real address of restart qu
           54   3C A7      D1     2502          cmpl    cddb$l_rstrtqfl(r7),r4   ; Empty restart queue
                   50      12     2506          bneq    30$                      ; branch if not empty
54   08 AC   0000004C 8F   C1     2508   5$:    addl3   #ucb$l_ioqfl,8(ap),r4    ; Get real address of queue header
           54   4C A2      D1     2511          cmpl    ucb$l_ioqfl(r2),r4       ; Empty i/o queue?
                   24      12     2515          bneq    7$                       ; Branch if not
      1F 64 A2   08        E0     2517          bbs     #ucb$v_bsy,ucb$w_sts(r2),7$   ; Branch if have IRP
           00000577'EF     95     251C          tstb    queue_notempty           ; if 0 all queues are empty
                   1A      12     2522          bneq    8$                       ; if 1 then at least 1 queue was not empty
                                  2524          skip    1
                                  252D          print   0,<!_*** I/O request queue is empty ***>
                           04     253A          ret
                                  253B
                 0033      31     253B   7$:    brw     50$                      ; process io request queue
                 0064      31     253E   8$:    brw     90$                      ; clear queue flag and return
                                  2541
                                  2541   ;      Queue - Class Driver Request Packet Queue (CDRP)
                                  2541   ;
              53   67      D0     2541   10$:   movl    cddb$l_cdrpqfl(r7),r3    ; Get address of first entry in queue
              56   01      D0     2544          movl    #1,r6                    ; Set state to current
           58   08 AC      D0     2547          movl    8(ap),r8                 ; pass actual address of ucb in r8
                 02D5      30     254B   20$:   bsbw    print_cdrp               ; display the contents of the cdrp
              53   65      D0     254E          movl    cdrp$l_fqfl(r5),r3       ; advance to next entry in queue
              54   53      D1     2551          cmpl    r3,r4                    ; check to see if another entry exists
                   A6      13     2554          beql    4$                       ; if points back to beginning no more
                   F3      11     2556          brb     20$                      ; process this entry in queue
                                  2558   ;
                                  2558   ;      Queue - Restarted Class Driver Request Packet Queue (RSTRTQ)
                                  2558   ;
           53   3C A7   D0  2558  2111 30$:      movl    cddb$l_rstrtqfl(r7),r3  ; Get first entry in queue
              56   02   D0  255C  2112           movl    #2,r6                   ; State is restart
           58   08 AC   D0  255F  2113           movl    8(ap),r8                ; pass actual address of ucb in r8
                 02BD   30  2563  2114 40$:      bsbw    print_cdrp              ; Call routine to display this cdrp
              53   65   D0  2566  2115           movl    cdrp$l_fqfl(r5),r3      ; Advance to next entry in queue
              54   53   D1  2569  2116           cmpl    r3,r4                   ; Check to see if no more entries in queue
                   F5   12  256C  2117           bneq    40$                     ; if eql branch to check next queue
                 FF97   31  256E  2118           brw     5$                      ; otherwise still more entries here to proce
                            2571  2119 ;
                            2571  2120 ;         Queue - Standard IO Request Packet Queue (IRP)
                            2571  2121 ;
         00000577'EF    95  2571  2122 50$:      tstb    queue_notempty          ; Check to see if anyone set this flag
                   0A   12  2577  2123           bneq    55$                     ; if 1 then yes so don't bother with it
                 03E6   30  2579  2124           bsbw    queue_title             ; print header for page (IO Request Queue)
     00000577'EF   01   90  257C  2125           movb    #1,queue_notempty       ; set flag to indicate queue was not empty
        0A 64 A2   08   E1  2583  2126 55$:      bbc     #ucb$v_bsy,ucb$w_sts(r2),60$     ; Branch if not busy
           53   58 A2   D0  2588  2127           movl    ucb$l_irp(r2),r3        ; Address of current IRP
              56   01   D0  258C  2128           movl    #1,r6                   ; Indicate current IRP
                 043E   30  258F  2129           bsbw    print_irp               ; Print line for current IRP
                            2592  2130
           53   4C A2   D0  2592  2131 60$:      movl    ucb$l_ioqfl(r2),r3      ; Get address of first IRP in queue
                   56   D4  2596  2132           clrl    r6                      ; Indicate not current IRP
                            2598  2133
              54   53   D1  2598  2134 70$:      cmpl    r3,r4                   ; end of queue?
                   08   13  259B  2135           beql    90$                     ; Branch if so
                 0430   30  259D  2136           bsbw    print_irp               ; print IRP line
              53   65   D0  25A0  2137           movl    irp$l_ioqfl(r5),r3      ; Skip to next IRP in queue
                   F3   11  25A3  2138           brb     70$
                            25A5  2139
         00000577'EF    94  25A5  2140 90$:      clrb    queue_notempty          ; clear flag before we are called again
                            25AB  2141           status  success
                        04  25B2  2142           ret
                            25B3  2143           .dsabl  lsb
```

```
                               25B3         .sbttl  show_acpq, display acp queue
                               25B3  ;---
                               25B3  ;
                               25B3  ;      show_acpq
                               25B3  ;
                               25B3  ;      Display the IRP queue associated with the ACP
                               25B3  ;      on the current volume.
                               25B3  ;
                               25B3  ; Inputs:
                               25B3  ;
                               25B3  ;      ap = address of VCB in local storage
                               25B3  ;
                               25B3  ;---
                               25B3         .enabl  lsb
                               25B3
                               25B3  show_acpq:
                     007C      25B3         .word   ^m<r2,r3,r4,r5,r6>
                               25B5
                  10 AC   D5   25B5         tstl    vcb$l_aqb(ap)           ; Is there any AQB?
                     03   12   25B8         bneq    10$                     ; Branch if so
                   0188   31   25BA  90$:   brw     95$                     ; Exit
                               25BD
52     0000041D'EF        9E   25BD  10$:   movab   aqb,r2
                               25C4         getmem  @vcb$l_aqb(ap),(r2),#aqb$c_length  ; Read entire AQB
                  E5 50   E9   25D2         blbc    r0,90$
                               25D5         ensure  11
                  10 AC   DD   25ED         pushl   vcb$l_aqb(ap)
                               25F0         skip    1
                               25F9         print   1,<!_!_    --- ACP Queue Block (AQB) !XL --->
                               2606         skip    1
                  0C A2   D5   260F         tstl    aqb$l_acppid(r2)        ; Is the XQP servicing this queue?
                     53   13   2612         beql    20$                     ; Branch if XQP
                               2614         getmem  @sch$gl_pcbvec,r3       ; Get address of PCB vector
                  4D 50   E9   2624         blbc    r0,30$
             51   0C A2   32   2627         cvtwl   aqb$l_acppid(r2),r1     ; Extract process index
             51    6341   DE   262B         moval   (r3)[r1],r1             ; Point to PCB address entry
                               262F         getmem  (r1)                    ; Read PCB address
                  39 50   E9   2638         blbc    r0,30$
53     00000000'EF        9E   263B         movab   buffer,r3
                               2642         getmem  pcb$t_lname(r1),(r3),#16  ; Read 16-byte process name
                  21 50   E9   2650         blbc    r0,30$
                  0C A2   DD   2653         pushl   aqb$l_acppid(r2)        ; Process PID
                     53   DD   2656         pushl   r3                      ; Address of ASCIC string
                               2658         print   1,<ACP requests are serviced by process !AC whose PID is !XL>
                     0D   11   2665         brb     30$
                               2667
                               2667  20$:   print   0,<ACP requests are serviced by the eXtended Qio Processor (XQP)>
                               2674
                               2674  30$:   skip    1
                               267D         alloc   80                      ; 80 byte string buffer
             7E   14 A2   9A   268C         movzbl  aqb$b_status(r2),-(sp)  ; ACP status
               E504 CF    9F   2690         pushab  acp_status              ; Bit definition table
00000000'EF          02   FB   2694         calls   #2,translate_bits       ; Translate bits into names
                     5E   DD   2698         pushl   sp                      ; Address of string descriptor
                  14 A2   DD   269D         pushl   aqb$b_status(r2)        ; ACP status
                               26A0         print   2,<Status: !XB !AS>
                               26AD         skip    1
```

```
                    26B6                 print_columns -
                    26B6                         (r2), vcb$l_aqb(ap), -
                    26B6                         aqb_column_T, aqb_column_2, aqb_column_3
                    26D4
                    26D4                 skip    1
         53   62 D0 26DD                 movl    aqb$l_acpqfl(r2),r3      ; Get address of first IRP
 54   10 AC   00 C1 26E0                 addl3   #aqb$l_acpqfl,vcb$l_aqb(ap),r4  ; Get real address of queuehead
         54   53 D1 26E5                 cmpl    r3,r4                    ; Empty ACP queue?
              0E 12 26E8                 bneq    70$                      ; Branch if not
                    26EA                 print   0,<!_*** ACP request queue is empty ***>
                 04 26F7                 ret
                    26F8
                    26F8    70$:         ensure  8
                    2710                 print   0,<!_!_!_!_ACP request queue>
                    271D                 print   0,<!_!_!_!_----------------->
                    272A                 skip    1
            026C 30 2733                 bsbw    irp_heading              ; Print heading line
              56 D4 2736                 clrl    r6                       ; Indicate not current IRP
                    2738
         54   53 D1 2738    80$:         cmpl    r3,r4                    ; End of queue?
              08 13 273B                 beql    95$                      ; Branch if so
            0290 30 273D                 bsbw    print_irp                ; Print IRP line
         53   65 D0 2740                 movl    irp$l_ioqfl(r5),r3       ; skip to next IRP
              F3 11 2743                 brb     80$
                    2745
                    2745    95$:         status  success
                 04 274C                 ret
                    274D                 .dsabl  lsb
                    274D                 .sbttl volume control block tables & action routines
                    274D
                    274D    ;  The following are all PRINT_COLUMNS action routines for the show_vcb
                    274D    ;  block displays.
                    274D    ;
                    274D    ;    Action Routine Inputs:
                    274D    ;
                    274D    ;        R2                value from the COLUMN_LIST entry
                    274D    ;        R5                size of value section for this item
                    274D    ;        R7                address of a descriptor for a scratch string in
                    274D    ;                          which the FAO converted value is to be returned
                    274D    ;        R11               base address of the local UCB copy
                    274D    ;
                    274D    ;    Action Routine Outputs:
                    274D    ;
                    274D    ;        R0                status
                    274D    ;                              lbs ==> use this entry
                    274D    ;                              lbc ==> skip this entry
                    274D    ;        R1 - R5           scratch
                    274D    ;                          all other registers must be preserved
                    274D
                    274D
                    274D    ;
                    274D    ; PRINT_COLUMNS tables for AQB display
                    274D    ;
                    274D
                    274D    aqb_column_1:
                    274D                 column_list -
                    274D                         aqb$, 16, 8, 4, < -
```

```
                           274D                      <<Mount count>,b_mntcnt,ub>, -
                           274D                      >
                           276D
                           276D      aqb_column_2:
                           276D              column_list -
                           276D                      aqb$, 16, 8, 4, < -
                           276D                      <<ACP type>,aqb_type,0,14,10>, -
                           276D                      <<ACP class>,aqb_class,0>, -
                           276D                      >
                           279D
                           279D      aqb_column_3:
                           279D              column_list -
                           279D                      aqb$, 16, 8, 0, < -
                           279D                      <<Linkage>,l_link,xl_neg>, -
                           279D                      <<Request queue>,l_acpqfl,q2>, -
                           279D                      >
                           27CD
                           27CD      ;***********
                           27CD      aqb_type:
      52    15 AB     9A   27CD              movzbl  aqb$b_acptype(r11), r2          ; get ACP type
     53   E3EB CF     9E   27D1              movab   aqb_acptype, r3                 ; get translate table
      00000000'GF     16   27D6              jsb     g^translate_address             ; translate ACP class
              0C      13   27DC              beql    90$                             ; branch if translate failed
         52   50      D0   27DE              movl    r0, r2                          ; setup translated string
                           27E1              do_column_entry ac, jmp                 ; display translation
                           27EA
      52    15 AB     9E   27EA      90$:    movab   aqb$b_acptype(r11), r2          ; else, get type address
                           27EE              do_column_entry ub, jmp                 ;  just display the value
                           27F7
                           27F7      ;***********
                           27F7      aqb_class:
      52    16 AB     9A   27F7              movzbl  aqb$b_class(r11), r2            ; get ACP class
              19      13   27FB              beql    90$                             ; branch if none
     53   D88F CF     9E   27FD              movab   ddb_acpclass, r3                ; get translate table
      00000000'GF     16   2802              jsb     g^translate_address             ; translate ACP class
              0C      13   2808              beql    90$                             ; branch if translate failed
         52   50      D0   280A              movl    r0, r2                          ; setup translated string
                           280D              do_column_entry ac, jmp                 ; display translation
                           2816
      52    13 AB     9E   2816      90$:    movab   ddb$b_acpclass(r11), r2         ; else, get class address
                           281A              do_column_entry ub, jmp                 ;  just display the value
```

```
                                  2823            .sbttl  print_cdrp, print a single CDRP block
                                  2823  ;---
                                  2823
                                  2823            .enabl  lsb
                                  2823  ;
                                  2823  ;         Subroutine to print information for a single CDRP block
                                  2823  ;
                                  2823  ; Inputs:
                                  2823  ;
                                  2823  ;         r3 = Dump address of CDRP block
                                  2823  ;         r6 = 2, if restarted CDRP, 1 if current CDRP
                                  2823  ;         r8 = Actual address of UCB
                                  2823  ;
                                  2823  ; Outputs:
                                  2823  ;
                                  2823  ;         r5 = Address of CDRP in local storage
                                  2823  ;
                                  2823  ;---
                                  2823
                                  2823  print_cdrp:
                                  2823            ensure  3
                      56     DD   283B            pushl   r6                        ; save r6
  56   53   FFFFFFA0  8F     C1   283D            addl3   #cdrp$l_ioqfl,r3,r6       ; get start of cdrp at most negative offset
       55   00000289'EF      9E   2845            movab   cdrp,r5                   ; get address of local cdrp
                                  284C            getmem  (r6),(r5),#cdrp_length    ; read entire CDRP
                      56   8ED0   285D            popl    r6                        ; restore r6
                   03 50     E8   2860            blbs    r0,5$                     ; check status
                    00FB     31   2863            brw     90$                       ; return
       55   FFFFFFA0  8F     C2   2866  5$:       subl2   #cdrp$l_ioqfl,r5          ; actual start of CDRP
             BC A5    58     D1   286D            cmpl    r8,cdrp$l_ucb(r5)         ; check to see if this request is from this
                      03     13   2871            beql    10$                       ; if equal yes so process it
                    00EB     31   2873            brw     90$                       ; return
            00000577'EF      95   2876  10$:      tstb    queue_notempty            ; Check to see if anyone set this flag
                      0A     12   287C            bneq    15$                       ; If 1 then yes so don't bother with it
                    00E1     30   287E            bsbw    queue_title               ; Otherwise display the header for page
         00000577'EF  01     90   2881            movb    #1,queue_notempty         ; set flag to say this queue was not empty
                   CA A5     DD   2888  15$:      pushl   cdrp$w_sts(r5)            ; request status
                   C4 A5     DD   288B            pushl   cdrp$l_iosb(r5)           ; address of IOSB
                   B0 A5     DD   288E            pushl   cdrp$l_ast(r5)            ; address of AST routine
                   C2 A5     DD   2891            pushl   cdrp$b_efn(r5)            ; Event flag number
                   B8 A5     DD   2894            pushl   cdrp$l_wind(r5)           ; Address of WCB
                   C0 A5     DD   2897            pushl   cdrp$w_func(r5)           ; Function code
                   C8 A5     DD   289A            pushl   cdrp$w_chan(r5)           ; Channel number
  50     AB A5    02  00     EF   289D            extzv   #irp$v_mode,#irp$s_mode,cdrp$b_rmod(r5),r0
              5553454B  8F     DD   28A3            pushl   #^a'KESU'                 ; Possible user modes
                    6E40     9F   28A9            pushab  (sp)[r0]                  ; Address of string
                      01     DD   28AC            pushl   #1                        ; Length of string
                   AC A5     DD   28AE            pushl   cdrp$l_pid(r5)            ; Process identification
                      53     DD   28B1            pushl   r3                        ; Address of CDRP
              00000043  8F     DD   28B3            pushl   #^a'C'                    ; String containing space
                      5E     DD   28B9            pushl   sp                        ; Address of string
                      01     DD   28BB            pushl   #1                        ; Length of string
                 56   01     D1   28BD            cmpl    #1,r6                     ; check if current CDRP
                      08     13   28C0            beql    20$                       ; branch if not
  08 AE     00000052  8F     D0   28C2            movl    #^a'R',8(sp)              ; Flag current CDRP being done
                                  28CA  20$:      print   15,< !AD!+    !XL  !XL  !AD!+  !XW  !XW  !XL  !20B  !XL  !XL  !XW>
                                  28D7
```

```
                              28D7  ; save a few registers now.  Then we will allocate stack space for two output
                              28D7  ; buffers.  Translate the class driver's flags field, the status field of the
                              28D7  ; cdrp, and the function code for the request.  Then display.
                              28D7  ;
             7E   52   7D     28D7                movq    r2, -(sp)                   ; save some registers
                              28DA                alloc   80,r2                       ; 80 byte output buffer for request status
                              28EC                alloc   80,r3                       ; another buffer of 80 bytes
          7E   40 A5   D0     28FE                movl    cdrp$l_dutuflags(r5),-(sp)  ; cdrp flags
             E0D2 CF   9F     2902                pushab  cdrp_dutuflags              ; bit definition table
      00000000'EF  02  FB     2906                calls   #2,translate_bits           ; translate bits to names
                  5E   DD     290D                pushl   sp                          ; push the address of descriptor
               04 A2   DD     290F                pushl   4(r2)                       ; push descriptor for request status
                  62   DD     2912                pushl   (r2)                        ; push size of this buffer
          7E   CA A5   3C     2914                movzwl  cdrp$w_sts(r5),-(sp)        ; request status
             E0F4 CF   9F     2918                pushab  request_status              ; bit definition table
      00000000'EF  02  FB     291C                calls   #2,translate_bits           ; translate bits to names
                  5E   DD     2923                pushl   sp                          ; address of string descriptor
         00000E21'EF   9F     2925                pushab  null_ascic                  ; assume function will not translate
 52   C0 A5   06   00  EF     292B                extzv   #io$v_fcode, #io$s_fcode, -
                              2931                        cdrp$w_func(r5), r2         ; get function code
        53   E153 CF   9E     2931                movab   io_function, r3             ; get translation table
         00000000'GF   16     2936                jsb     g^translate_address         ; translate function to text
                  03   13     293C                beql    33$                         ; branch if translate failed
             6E   50   D0     293E                movl    r0, (sp)                    ; setup translated function
                              2941       33$:     print   3,<!_!AC !AS!+!+ !AS>       ; print translated information
                              294E                skip    1                           ; advance
    5E   000000B8 8F   C0     2957                addl    #184,sp                     ; deallocate translate buffers
             52   8E   7D     295E                movq    (sp)+, r2                   ; restore saved registers
                              2961
                       05     2961       90$:     rsb
```

```
                                  2962            .sbttl  print_irp, print a single IRP block
                                  2962    ;---
                                  2962
                                  2962            .enabl  lsb
                                  2962    ;
                                  2962    ;       Subroutine to print information for a single IRP block
                                  2962    ;
                                  2962    ; Inputs:
                                  2962    ;
                                  2962    ;       r3 = Dump address of IRP block
                                  2962    ;       r6 = 1 if current IRP, else 0 if pending
                                  2962    ;
                                  2962    ; Outputs:
                                  2962    ;
                                  2962    ;       r5 = Address of IRP in local storage
                                  2962    ;
                                  2962    ;---
                                  2962
                                  2962    queue_title:
                                  2962            ensure  10
                                  297A            print   0,<!_!_!_!_I/O request queue>
                                  2987            print   0,<!_!_!_!_----------------->
                                  2994            skip    1
     17 3C A2    05      E0       299D            bbs     #dev$v_mscp,ucb$l_devchar2(r2),cdrp_heading
                                  29A2    irp_heading:
                                  29A2            print   0,<STATE    IRP      PID    MODE CHAN  FUNC    WCB      EFN    AST
                                  29AF            skip    1
                 05               29B8    999$:   rsb
                                  29B9
                                  29B9    cdrp_heading:
                                  29B9            print   0,<STATE CDRP/IRP    PID    MODE CHAN  FUNC    WCB      EFN    AST
                                  29C6            skip    1
                 05               29CF            rsb
                                  29D0
                                  29D0    print_irp:
                                  29D0            ensure  3
 55    000001C5'EF       9E       29E8            movab   irp,r5
                                  29EF            getmem  (r3),(r5),#irp$c_length ; read entire IRP
            B5 50        E9       2A00            blbc    r0,999$
            2A A5        DD       2A03            pushl   irp$w_sts(r5)                   ; request status
            24 A5        DD       2A06            pushl   irp$l_iosb(r5)                  ; address of IOSB
            10 A5        DD       2A09            pushl   irp$l_ast(r5)                   ; address of AST routine
            22 A5        DD       2A0C            pushl   irp$b_efn(r5)                   ; Event flag number
            18 A5        DD       2A0F            pushl   irp$l_wind(r5)                  ; Address of WCB
            20 A5        DD       2A12            pushl   irp$w_func(r5)                  ; Function code
            28 A5        DD       2A15            pushl   irp$w_chan(r5)                  ; Channel number
50   0B A5   02   00     EF       2A18            extzv   #irp$v_mode,#irp$s_mode,irp$b_rmod(r5),r0
        5553454B 8F      DD       2A1E            pushl   #^a'KESU'                       ; Possible user modes
            6E40         9F       2A24            pushab  (sp)[r0]                        ; Address of string
              01         DD       2A27            pushl   #1                              ; Length of string
            0C A5        DD       2A29            pushl   irp$l_pid(r5)                   ; Process identification
              53         DD       2A2C            pushl   r3                              ; Address of IRP
        00000050 8F      DD       2A2E            pushl   #^a'P'                          ; String containing space
              5E         DD       2A34            pushl   sp                              ; Address of string
              01         DD       2A36            pushl   #1                              ; Length of string
              56         D5       2A38            tstl    r6                              ; check if current IRP
              08         13       2A3A            beql    20$                             ; branch if not
```

```
       08 AE    00000043 8F   D0  2A3C              movl    #^a'C',8(sp)              ; Flag current IRP being done
                                  2A44       20$:   print   15,< !AD!+   !XL  !XL  !AD!+   !XW  !XW  !XL  !2UB  !XL  !XL  !XW>
                                  2A51
                 7E   52      7D  2A51              movq    r2, -(sp)                 ; save some registers
                                  2A54              alloc   80                        ; 80 byte output buffer
            7E    2A A5       3C  2A63              movzwl  irp$w_sts(r5),-(sp)       ; request status
                    DFA5 CF   9F  2A67              pushab  request_status            ; bit definition table
         00000000'EF      02  FB  2A6B              calls   #2,translate_bits         ; translate bits to names
                          5E  DD  2A72              pushl   sp                        ; address of string descriptor
              00000E21'EF     9F  2A74              pushab  null_ascic                ; assume function will not translate
   52    20 A5    06      00  EF  2A7A              extzv   #io$v_fcode, #io$s_fcode, -
                                  2A80                      irp$w_func(r5), r2        ; get function code
           53    E004 CF      9E  2A80              movab   io_function, r3           ; get translation table
              00000000'GF     16  2A85              jsb     g^translate_address       ; translate function to text
                          03  13  2A8B              beql    33$                       ; branch if translate failed
                 6E   50      D0  2A8D              movl    r0, (sp)                  ; setup translated function
                                  2A90       33$:   print   2,<!_!AC !AS>             ; print translated information
                                  2A9D              skip    1
          5E    00000058 8F   C0  2AA6              addl    #88,sp                    ; deallocate translate buffer
                 52   8E      7D  2AAD              movq    (sp)+, r2                 ; restore saved registers
                                  2AB0
                              05  2AB0       90$:   rsb
                                  2AB1
                                  2AB1              .dsabl  lsb
```

```
                             2AB1          .sbttl  show_vcb, Display Volume Control Block (VCB)
                             2AB1  ;---
                             2AB1  ;
                             2AB1  ;       show_vcb
                             2AB1  ;
                             2AB1  ;       Display the Volume Control Block (VCB)
                             2AB1  ;
                             2AB1  ; Inputs:
                             2AB1  ;
                             2AB1  ;       ap = Address of UCB in local storage
                             2AB1  ;
                             2AB1  ;---
                             2AB1
                             2AB1  show_vcb:
                     083C    2AB1          .word   ^m<r2,r3,r4,r5,r11>
                             2AB3
              34 AC   D5     2AB3          tstl    ucb$l_vcb(ap)           ; any VCB for this unit?
                 08   12     2AB6          bneq    10$                     ; Branch if so
                             2AB8
                             2AB8  90$:    status  success
                      04     2ABF          ret
                             2AC0
     F3 38 AC    06   E0     2AC0  10$:    bbs     #dev$v_spl,ucb$l_devchar(ap),90$  ; ignore VCB for
                             2AC5                                          ; spooled devices (wrong usage)
  52   00000331'EF    9E     2AC5          movab   vcb,r2
                             2ACC          getmem  @ucb$l_vcb(ap),(r2),#vcb$c_length ; read entire VCB
              D7 50   E9     2ADE          blbc    r0,90$
        11    0A A2   91     2AE1          cmpb    vcb$b_type(r2),#dyn$c_vcb       ; Check if block valid
                 D1   12     2AE5          bneq    90$                     ; Exit if not valid type
                             2AE7
                             2AE7          ensure  12
                             2AFF          skip    1
              34 AC   DD     2B08          pushl   ucb$l_vcb(ap)
                             2B0B          print   1,<!_T_--- Volume Control Block (VCB) !XL --->
                             2B18          skip    1
                             2B21          alloc   80                      ; 80 byte output buffer
              5B   5E   D0   2B30          movl    sp, r11                 ; save descriptor address
                             2B33
                             2B33          ; use different display stratagies for different VCB types
     03 38 AC    0D   E1     2B33          bbc     #dev$v_net, ucb$l_devchar(ap), 20$
               028D   31     2B38          brw     vcb_net
                             2B3B  20$:    dispatch ucb$b_devclass(ap), type=B, prefix=dc$_, <-
                             2B3B                  <disk,vcb_disk>, -
                             2B3B                  <tape,vcb_tape>, -
                             2B3B                  <journal,vcb_journal> -
                             2B3B                  >
                             2C83          status  success
                      04     2C8A          ret
                             2C8B
                             2C8B  vcb_disk:
     03 38 AC    18   E1     2C8B          bbc     #dev$v_for, -           ; Is this a foreign mounted disk?
                             2C90                  ucb$l_devchar(ap), 20$
               008A   31     2C90          brw     vcb_foreign             ; Branch if foreign.
            0080 C2   DF     2C93  20$:    pushal  vcb$t_volcknam(r2)      ; Address of volume lock name
                 0C   DD     2C97          pushl   #12                     ; Length of volume lock name
              14 A2   DF     2C99          pushal  vcb$t_volname(r2)       ; Address of volume name
                 0C   DD     2C9C          pushl   #12                     ; Length of volume name
```

```
                             2C9E                print   2,<Volume: !AD     Lock name: !AF>
          5E    5B      D0   2CAB                movl    r11, sp                ; Setup scratch area
     7E    0B   A2      9A   2CAE                movzbl  vcb$b_status(r2), -(sp) ; Volume status
            DB22 CF     9F   2CB2                pushab  vcb_disk_status        ; Bit definition table
  00000000'EF   02      FB   2CB6                calls   #2, translate_bits     ; Translate bits to names
                5E      DD   2CBD                pushl   sp                     ; Address of output descriptor
            0B  A2      DD   2CBF                pushl   vcb$b_status(r2)
                             2CC2                print   2,<Status:  !XB !AS>
          5E    5B      D0   2CCF                movl    r11, sp                ; Setup scratch area
     7E    53   A2      9A   2CD2                movzbl  vcb$b_status2(r2), -(sp); Volume status, second byte
            DB46 CF     9F   2CD6                pushab  vcb_disk_status2       ; Bit definition table
  00000000'EF   02      FB   2CDA                calls   #2, translate_bits     ; Translate bits to names
                5E      DD   2CE1                pushl   sp                     ; Address of output descriptor
            53  A2      DD   2CE3                pushl   vcb$b_status2(r2)
                             2CE6                print   2,<Status2: !XB !AS>
                             2CF3                skip    1
                             2CFC                print_columns -
                             2CFC                        (r2), ucb$l_vcb(ap), -
                             2CFC                        vcb_disk_col_1, vcb_disk_col_2, vcb_disk_col_3
             0129       31   2D1A                brw     vcb_show_acpq
                             2D1D
                             2D1D                .enable lsb
                             2D1D      vcb_tape:
                             2D1D      vcb_foreign:
            14  A2      DF   2D1D                pushal  vcb$t_volname(r2)      ; Address of volume name
                0C      DD   2D20                pushl   #12                    ; Length of volume name
                             2D22                print   1,<Volume: !AD>
    22 3B AC    18      E1   2D2F                bbc     #dev$v_for, -          ; Is this a foreign mounted volume?
                             2D34                        ucb$l_devchar(ap), 20$ ; Branch if not foreign.
                             2D34                skip    1
                             2D3D                print   0,<!_!_!_Volume is foreign mounted>
                             2D4A                skip    1
             00F0       31   2D53                brw     vcb_show_acpq          ; Go try to do AQB, ha ha.
          5E    5B      D0   2D56      20$:      movl    r11, sp                ; Setup scratch area
     7E    0B   A2      9A   2D59                movzbl  vcb$b_status(r2), -(sp) ; Volume status
            DAEF CF     9F   2D5D                pushab  vcb_tape_status        ; Bit definition table
  00000000'EF   02      FB   2D61                calls   #2, translate_bits     ; Translate bits to names
                5E      DD   2D68                pushl   sp                     ; Address of output descriptor
            0B  A2      DD   2D6A                pushl   vcb$b_status(r2)
                             2D6D                print   2,<Status: !4XB !AS>
          5E    5B      D0   2D7A                movl    r11, sp                ; Setup scratch area
     7E    2C   A2      3C   2D7D                movzwl  vcb$w_mode(r2), -(sp)  ; Volume operating mode
            DB13 CF     9F   2D81                pushab  vcb_tape_mode          ; Bit definition table
  00000000'EF   02      FB   2D85                calls   #2, translate_bits     ; Translate bits to names
                5E      DD   2D8C                pushl   sp                     ; Address of output descriptor
            2C  A2      DD   2D8E                pushl   vcb$w_mode(r2)
                             2D91                print   2,<Mode:   !4XW !AS>
                             2D9E                skip    1
                             2DA7                print_columns -
                             2DA7                        (r2), ucb$l_vcb(ap), -
                             2DA7                        vcb_tape_col_1, vcb_tape_col_2, vcb_tape_col_3
             007E       31   2DC5                brw     vcb_show_acpq
                             2DC8
                             2DC8                .disable lsb
                             2DC8
                             2DC8      vcb_net:
                             2DC8                print_columns -
```

```
                           2DC8                   (r2), ucb$l_vcb(ap), -
                           2DC8                   vcb_net_col_1, vcb_net_col_2, vcb_net_col_3
          005D    31       2DE6           brw     vcb_show_acpq
                           2DE9
                           2DE9  vcb_journal:
         14 A2    DF       2DE9           pushal  vcb$t_volname(r2)          ; Address of journalname
            0C    DD       2DEC           pushl   #12                        ; Length of journal name
                           2DEE           print   1,<Journal name: !AD>
     5E     5B    D0       2DFB           movl    r11, sp                    ; Setup scratch area
  7E     24 A2    D0       2DFE           movl    vcb$l_jnl_char(r2),-(sp); Journal characteristics
       DB02 CF    9F       2E02           pushab  vcb_journal_char           ; Bit definition table
00000000'EF  02   FB       2E06           calls   #2, translate_bits         ; Translate bits to names
            5E    DD       2E0D           pushl   sp                         ; Address of output descriptor
         24 A2    DD       2E0F           pushl   vcb$l_jnl_char(r2)
                           2E12           print   2,<Characteristics: !XL !AS>
                           2E1F           skip    1
                           2E28           print_columns -
                           2E28                   (r2), ucb$l_vcb(ap), -
                           2E28                   vcb_jnl_col_1, vcb_jnl_col_2, vcb_jnl_col_3
                           2E46
                           2E46  vcb_show_acpq:
    F768 CF   62    FA     2E46           callg   (r2),show_acpq             ; Display ACP queue (if any)
                    04     2E4B           ret
```

```
2E4C          .sbttl volume control block tables & action routines
2E4C
2E4C   ; The following are all PRINT_COLUMNS action routines for the show_vcb
2E4C   ; block displays.
2E4C   ;
2E4C   ;   Action Routine Inputs:
2E4C   ;
2E4C   ;      R2             value from the COLUMN_LIST entry
2E4C   ;      R5             size of value section for this item
2E4C   ;      R7             address of a descriptor for a scratch string in
2E4C   ;                     which the FAO converted value is to be returned
2E4C   ;      R11            base address of the local UCB copy
2E4C   ;
2E4C   ;   Action Routine Outputs:
2E4C   ;
2E4C   ;      R0             status
2E4C   ;                         lbs ==> use this entry
2E4C   ;                         lbc ==> skip this entry
2E4C   ;      R1 - R5        scratch
2E4C   ;                     all other registers must be preserved
2E4C
2E4C
2E4C  ;
2E4C  ; PRINT_COLUMNS tables for disk VCB displays
2E4C  ;
2E4C
2E4C  vcb_disk_col_1:
2E4C          column_list -
2E4C                  vcb$, 16, 8, 4, < -
2E4C                  <<Mount count>,w_mcount,uw>, -
2E4C                  <<Transactions>,w_trans,uw>, -
2E4C                  <<Free blocks>,l_free,ul>, -
2E4C                  <<Window size>,b_window,ub>, -
2E4C                  <<Vol. lock ID>,l_vollkid,xl_neq>, -
2E4C                  <<Block. lock ID>,l_blockid,xl_neq>, -
2E4C                  >
2EBC
2EBC  vcb_disk_col_2:
2EBC          column_list -
2EBC                  vcb$, 16, 8, 4, < -
2EBC                  <<Rel. volume>,w_rvn,uw>, -
2EBC                  <<Max. files>,l_maxfiles,ul>, -
2EBC                  <<Rsvd. files>,b_resfiles,ub>, -
2EBC                  <<Cluster size>,w_cluster,uw>, -
2EBC                  <<Def. extend sz.>,w_extend,uw>, -
2EBC                  <<Record size>,w_recordsz,uw>, -
2EBC                  >
2F2C
2F2C  vcb_disk_col_3:
2F2C          column_list -
2F2C                  vcb$, 16, 8, 0, < -
2F2C                  <<AQB address>,l_aqb,xl>, -
2F2C                  <<RVT address>,l_rvt,xl>, -
2F2C                  <<FCB queue>,l_fcbfl,q2>, -
2F2C                  <<Quota FCB>,l_quotafcb,xl_neq>, -
2F2C                  <<Quota cache>,l_quocache,xl_neq>, -
2F2C                  <<Cache blk.>,l_cache,xl_neq>, -
```

```
2F2C                    >
2F9C
2F9C    ;
2F9C    ; PRINT_COLUMNS tables for tape VCB displays
2F9C    ;
2F9C
2F9C    vcb_tape_col_1:
2F9C            column_list -
2F9C                    vcb$, 16, 8, 4, < -
2F9C                    <<Transactions>,w_trans,uw>, -
2F9C                    <<Start record>,l_st_record,ul>, -
2F9C                    <<Tapemark count>,b_tm,ub>, -
2F9C                    >
2FDC
2FDC    vcb_tape_col_2:
2FDC            column_list -
2FDC                    vcb$, 16, 8, 4, < -
2FDC                    <<Rel. volume>,b_cur_rvn,ub>, -
2FDC                    <<Tape vol. list>,l_mvl,xl_neq>, -
2FDC                    >
300C
300C    vcb_tape_col_3:
300C            column_list -
300C                    vcb$, 16, 8, 0, < -
300C                    <<AQB address>,l_aqb,xl>, -
300C                    <<Virt. pg. queue>,l_vpfl,q2>, -
300C                    <<Blocked queue>,l_blockfl,q2>, -
300C                    >
304C
304C    ;
304C    ; PRINT_COLUMNS tables for network VCB displays
304C    ;
304C
304C    vcb_net_col_1:
304C            column_list -
304C                    vcb$, 16, 8, 4, < -
304C                    <<Transactions>,w_trans,uw>, -
304C                    >
306C
306C    vcb_net_col_2:
306C            column_list -
306C                    vcb$, 16, 8, 4, < -
306C                    <<Mount count>,w_mcount,uw>, -
306C                    >
308C
308C    vcb_net_col_3:
308C            column_list -
308C                    vcb$, 16, 8, 0, < -
308C                    <<AQB address>,l_aqb,xl>, -
308C                    >
30AC
30AC    ;
30AC    ; PRINT_COLUMNS tables for journal VCB displays
30AC    ;
30AC
30AC    vcb_jnl_col_1:
30AC            column_list -
```

```
30AC                          vcb$, 16, 8, 4, < -
30AC                          <<Copies>,w_jnl_cop,uw>, -
30AC                          <<Mask>,l_jnl_mask,xl>, -
30AC                          >
30DC
30DC   vcb_jnl_col_2:
30DC                  column_list -
30DC                          vcb$, 16, 8, 4, < -
30DC                          <<Access mode>,b_jnl_mode,xb>, -
30DC                          <<JFT address>,l_jnl_jfta,xl>, -
30DC                          >
310C
310C   vcb_jnl_col_3:
310C                  column_list -
310C                          vcb$, 16, 8, 0, < -
310C                          <<AQB address>,l_aqb,xl>, -
310C                          <<JMT address>,l_jnl_jmt,xl>, -
310C                          >
```

```
                            313C              .sbttl  show_cddb, Display Class Driver Data Block (CDDB)
                            313C  ;---
                            313C  ;
                            313C  ;           show_cddb
                            313C  ;
                            313C  ;           Display the Class Driver Data Block (CDDB)
                            313C  ;
                            313C  ; Inputs:
                            313C  ;
                            313C  ;           ap = Address of UCB in local storage
                            313C  ;           r6 = actual address of cddb
                            313C  ;
                            313C  ;---
                            313C
                            313C  show_cddb:
                     083C   313C              .word   ^m<r2,r3,r4,r5,r11>
                            313E
                  56   D5   313E              tstl    r6                          ; is there a cddb
                  23   13   3140              beql    5$                          ; no, so exit
  52    00000471'EF    9E   3142              movab   cddb, r2                    ; store address of local cddb
                            3149              getmem  (r6),(r2),#cddb$c_length            ; read entire cddb
               08 50   E9   315A              blbc    r0,5$                       ; return if not able to read it
                            315D
   0A A2    0164 8F    B1   315D              cmpw    #<dyn$c_cd_cddb@8+dyn$c_classdrv>,cddb$b_type(r2)
                            3163                                                  ; check for valid block type
                  0B   13   3163              beql    10$                         ; exit if not valid type
                            3165  5$:
                            3165              status  success                     ;
                       04   316C              ret
                            316D
                            316D  10$:        ensure  20                          ; need 15 lines for this display
                            3185              skip    1                           ; advance 1 line
                  56   DD   318E              pushl   r6                          ; pass address of cddb to print routine
         00000575'EF   B5   3190              tstw    flag_2nd_cddb               ; 0 - primary, 1 - secondary
                  0F   12   3196              bneq    second                      ; secondary if branch
                            3198              print   1,<!_!_--- Primary Class Driver Data Block (CDDB) !XL --->
                  0D   11   31A5              brb     display
                            31A7  second:
                            31A7              print   1,<!_!_--- Secondary Class Driver Data Block (CDDB) !XL --->
                            31B4  display:
                            31B4              skip    1                           ; advance 1 line
           5B     5E   D0   31BD              movl    sp,r11                      ; save pre-allocation stack pointer
                            31C0              alloc   80,r4                       ; 80 byte output buffer
     7E     12 A2      3C   31D2              movzwl  cddb$w_status(r2),-(sp)     ; cddb status field
          D74E CF      9F   31D6              pushab  cddb_status                 ; bit definition table
  00000000'EF   02     FB   31DA              calls   #2,translate_bits           ; translate bits to names
                54     DD   31E1              pushl   r4                          ; address of output descriptor
     7E     12 A2      3C   31E3              movzwl  cddb$w_status(r2),-(sp)     ; pass value of status field to print
                            31E7              print   2,<Status:                  !XW !AS>          ; display status
     64     50 8F      9A   31F4              movzbl  #80,(r4)                    ;
     7E     28 A2      3C   31F8              movzwl  cddb$w_cntrlflgs(r2),-(sp)  ; cddb controller flags
          D790 CF      9F   31FC              pushab  cddb_flags                  ; bit definition table
  00000000'EF   02     FB   3200              calls   #2,translate_bits           ; translate bits to names
                54     DD   3207              pushl   r4                          ; address of output descriptor
     7E     28 A2      3C   3209              movzwl  cddb$w_cntrlflgs(r2),-(sp)        ; pass value of status field to prin
                            320D              print   2,<Controller Flags:    !XW !AS>  ; display status
           5E     5B   D0   321A              movl    r11,sp                      ; restore stack pointer
```

```
      321D          skip    1                               ; advance 1 line
      3226          print_columns   -
      3226                  (r2), r6,-
      3226                  cddb_col_1,cddb_col_2,cddb_col_3        ;display!!!!
      3243          status  success
  04  324A          ret
      324B
      324B
```

```
3248                   .sbttl class driver data block tables & action routines
3248
3248       ;  The following are all PRINT_COLUMNS action routines for the show_cddb
3248       ;  block displays.
3248       ;
3248       ;    Action Routine Inputs:
3248       ;
3248       ;       R2              value from the COLUMN_LIST entry
3248       ;       R5              size of value section for this item
3248       ;       R7              address of a descriptor for a scratch string in
3248       ;                       which the FAO converted value is to be returned
3248       ;       R11             base address of the local UCB copy
3248       ;
3248       ;    Action Routine Outputs:
3248       ;
3248       ;       R0              status
3248       ;                           lbs ==> use this entry
3248       ;                           lbc ==> skip this entry
3248       ;       R1 - R5         scratch
3248       ;                       all other registers must be preserved
3248
3248
3248       ;
3248       ; PRINT_COLUMNS tables for CDDB displays
3248       ;
3248
3248       cddb_col_1:
3248               column_list -
3248                       cddb$, 16, 8, 4, < -
3248                       <<Allocation class>,l_allocls,ul>, -
3248                       <<System ID>,cddb_4bytes,cddb$b_systemid>, -
3248                       <<>,cddb_2bytes,cddb$b_systemid+4>,-
3248                       <<Contrl. ID>,cddb_4bytes,cddb$q_cntrlid>, -
3248                       <<>,cddb_4bytes,cddb$q_cntrlid+4>,-
3248                       <<Response ID>,l_oldrspid,xl>, -
3248                       <<MSCP Cmd status>,l_oldcmdsts,xl>,-
3248                       >
32CB
32CB       cddb_col_2:
32CB               column_list -
32CB                       cddb$, 16, 8, 4, < -
32CB                       <<CDRP Queue>,l_cdrpqfl,q2>, -
32CB                       <<Restart Queue>,l_rstrtqfl,q2>, -
32CB                       <<Restarted CDRP>,rstrt_cdrp,cddb$l_rstrtcdrp>, -
32CB                       <<CDRP retry cnt.>,retry_cnt,cddb$b_retrycnt>, -
32CB                       <<DAP Count>,b_dapcount,ub>, -
32CB                       <<Contr. timeout>,w_cntrltmo,uw>, -
32CB                       <<Reinit Count>,w_rstrtcnt,uw>, -
32CB                       <<Wait UCB Count>,w_wtucbctr,uw>, -
32CB                       >
335B
335B
335B       cddb_col_3:
335B               column_list -
335B                       cddb$, 16, 8, 0, < -
335B                       <<DDB address>,l_ddb,xl>, -
335B                       <<CRB address>,l_crb,xl>, -
```

```
                 335B                  <<CDDB link>,l_cddblink,xl>, -
                 335B                  <<PDT address>,l_pdt,xl>, -
                 335B                  <<Original UCB>,l_origucb,xl>,-
                 335B                  <<UCB chain>,l_ucbchain,xl>, -
                 335B                  >
                 33CB
                 33CB  ;**********
                 33CB  cddb_4bytes:
  53  5B  52  C1 33CB          addl3   r2,r11,r3                        ; locate storage of interest
      55  08  C2 33CF          subl    #8,r5                            ; get size of filler field
                 33D2          $fao_s  -
                 33D2                  ctrstr=cddb_fao,-
                 33D2                  outbuf = (r7),-
                 33D2                  outlen = (r7),-
                 33D2                  p1 = r5,-
                 33D2                  p2 = (r3)
              05 33E7          rsb
                 33E8
                 33E8  ;**********
                 33E8  cddb_2bytes:
  53  5B  52  C1 33E8          addl3   r2, r11, r3                      ; locate storage of interest
      55  04  C2 33EC          subl    #4, r5                           ; get size of filler field
                 33EF          $fao_s  -
                 33EF                  ctrstr = sb_fao_6bytes, -
                 33EF                  outbuf = (r7), -
                 33EF                  outlen = (r7), -
                 33EF                  p1 = r5, -
                 33EF                  p2 = (r3)
              05 3404          rsb
                 3405
                 3405  ;**********
                 3405  rstrt_cdrp:
0C 12 AB  00  E1 3405          bbc     #cddb$v_snglstrm,cddb$w_status(r11),cddb_act_nop
                 340A                                          ; cdrp only exists if single stream
      52  5B  C0 340A          addl    r11,r2                  ; locate cell to return
                 340D          do_column_entry xl,jmp          ; display this entry
                 3416
                 3416  cddb_act_nop:
      50  D4     3416          clrl    r0
          05     3418          rsb
                 3419
                 3419  ;**********
                 3419  retry_cnt:
F8 12 AB  00  E1 3419          bbc     #cddb$v_snglstrm,cddb$w_status(r11),cddb_act_nop
                 341E                                          ; count is valid if single stream
      52  5B  C0 341E          addl    r11,r2                  ; locate cell to return
                 3421          do_column_entry ub,jmp          ; display this entry
                 342A
```

```
                                  342A  2914          .end
```

```
$$$                              = 00000871 R     04     CDDB$L_ALLOCLS                  = 00000050
$$.TMP1                          = 00000001              CDDB$L_CDDBLINK                 = 00000058
$$.TMP2                          = 000000EF              CDDB$L_CDRPQFL                  = 00000000
$$BASE                           = 00000001              CDDB$L_CRB                      = 00000018
$$DISPL                          = 000000A2              CDDB$L_DDB                      = 0000001C
$$GENSW                          = 00000001              CDDB$L_OLDCMDSTS                = 00000030
$$HIGH                           = 000000A1              CDDB$L_OLDRSPID                 = 0000002C
$$LIMIT                          = 000000A0              CDDB$L_ORIGUCB                  = 0000004C
$$LOW                            = 00000001              CDDB$L_PDT                      = 00000014
$$MNSW                           = 00000001              CDDB$L_RSTRTCDRP                = 00000034
$$MXSW                           = 00000001              CDDB$L_RSTRTQFL                 = 0000003C
$$T2                             = 00000005              CDDB$L_UCBCHAIN                 = 00000048
ACP_STATUS                         00000B98 R     03     CDDB$Q_CNTRLID                  = 00000020
ADD_SYMBOL                         ********   X   03     CDDB$V_2PBSY                    = 0000000B
ADP$W_ADPTYPE                    = 0000000E              CDDB$V_ALCLS_SET                = 00000006
AQB                                0000041D R     02     CDDB$V_DAPBSY                   = 0000000A
AQB$B_ACPTYPE                    = 00000015              CDDB$V_IMPEND                   = 00000001
AQB$B_CLASS                      = 00000016              CDDB$V_INITING                  = 00000002
AQB$B_MNTCNT                     = 0000000B              CDDB$V_NOCONN                   = 00000007
AQB$B_STATUS                     = 00000014              CDDB$V_POLLING                  = 00000005
AQB$C_LENGTH                     = 0000001C              CDDB$V_QUORLOST                 = 00000009
AQB$K_F11V1                      = 00000001              CDDB$V_RECONNECT                = 00000003
AQB$K_F11V2                      = 00000002              CDDB$V_RESYNCH                  = 00000004
AQB$K_JNL                        = 00000006              CDDB$V_RSTRTWAIT                = 00000008
AQB$K_MTA                        = 00000003              CDDB$V_SNGLSTRM                 = 00000000
AQB$K_NET                        = 00000004              CDDB$W_CNTRLFLGS                = 00000028
AQB$K_REM                        = 00000005              CDDB$W_CNTRLTMO                 = 0000002A
AQB$K_UNDEFINED                  = 00000000              CDDB$W_RSTRTCNT                 = 0000003A
AQB$L_ACPPID                     = 0000000C              CDDB$W_STATUS                   = 00000012
AQB$L_ACPQFL                     = 00000000              CDDB$W_WTUCBCTR                 = 0000005E
AQB$L_LINK                       = 00000010              CDDB_2BYTES                       000033E8 R     03
AQB$V_CREATING                   = 00000003              CDDB_2P                           000004E1 R     02
AQB$V_DEFCLASS                   = 00000001              CDDB_4BYTES                       000033CB R     03
AQB$V_DEFSYS                     = 00000002              CDDB_ACT_NOP                      00003416 R     03
AQB$V_UNIQUE                     = 00000000              CDDB_COL_1                        0000324B R     03
AQB_ACPTYPE                        00000BC0 R     03     CDDB_COL_2                        000032CB R     03
AQB_CLASS                          000027F7 R     03     CDDB_COL_3                        0000335B R     03
AQB_COLUMN_1                       0000274D R     03     CDDB_FAO                          00000E12 R     04
AQB_COLUMN_2                       0000276D R     03     CDDB_FLAGS                        00000990 R     03
AQB_COLUMN_3                       0000279D R     03     CDDB_STATUS                       00000928 R     03
AQB_TYPE                           000027CD R     03     CDRP                              00000289 R     02
ARGS                             = 00000003              CDRP$B_EFN                      = FFFFFFC2
AT$_UBA                          = 00000001              CDRP$B_RMOD                     = FFFFFFAB
BAD_ASCIC                          00000E36 R     04     CDRP$C_CD_LEN                   = 00000048
BIT...                           = 00000003              CDRP$L_AST                      = FFFFFFB0
BUFFER                             ********   X   03     CDRP$L_DUTUFLAGS                = 00000040
BUS_TYPE                           00000728 R     03     CDRP$L_FQFL                     = 00000000
CARD_TYPE                          00000548 R     03     CDRP$L_IOQFL                    = FFFFFFA0
CBL_A_ASCIC                        00000E2C R     04     CDRP$L_IOSB                     = FFFFFFC4
CBL_B_ASCIC                        00000E2F R     04     CDRP$L_PID                      = FFFFFFAC
CDDB                               00000471 R     02     CDRP$L_UCB                      = FFFFFFBC
CDDB$B_DAPCOUNT                  = 00000039              CDRP$L_WIND                     = FFFFFFB8
CDDB$B_RETRYCNT                  = 00000038              CDRP$V_CAND                     = 00000000
CDDB$B_SYSTEMID                  = 0000000C              CDRP$V_CANIO                    = 00000001
CDDB$B_TYPE                      = 0000000A              CDRP$V_ERLIP                    = 00000002
CDDB$C_LENGTH                    = 00000070              CDRP$V_HIRT                     = 00000004
CDDB$K_LENGTH                    = 00000070              CDRP$V_IVCMD                    = 00000008
```

```
CDRP$V_PERM                      = 00000003                DDB$L_ACPD                       = 00000010
CDRP$W_CHAN                      = FFFFFFC8                DDB$L_ALLOCLS                    = 0000003C
CDRP$W_FUNC                      = FFFFFFC0                DDB$L_CONLINK                    = 00000038
CDRP$W_STS                       = FFFFFFCA                DDB$L_DDT                        = 0000000C
CDRP_DOTUFLAGS                     000009D8 R      03      DDB$L_DP_UCB                     = 00000040
CDRP_HEADING                       000029B9 R      03      DDB$L_LINK                       = 00000000
CDRP_LENGTH                      = 000000A8                DDB$L_SB                         = 00000034
CMND_BUFFER                        ********    X   03      DDB$L_UCB                        = 00000004
CMND_DESCR                         ********    X   03      DDB$T_DRVNAME                    = 00000024
COLM$K_FAO_AC                    = 00000000                DDB$T_NAME                       = 00000014
COLM$K_FAO_AS                    = 00000001                DDB_2P                             000000B5 R      02
COLM$K_FAO_OW                    = 00000007                DDB_ACPCLASS                       00000090 R      03
COLM$K_FAO_Q2                    = 00000011                DDB_ACPCLS                         000013C4 R      03
COLM$K_FAO_UB                    = 00000005                DDB_ACPD                           0000139E R      03
COLM$K_FAO_UL                    = 0000000F                DDB_COLUMN_1                       000012DE R      03
COLM$K_FAO_UW                    = 0000000A                DDB_COLUMN_2                       0000131E R      03
COLM$K_FAO_XB                    = 00000003                DDB_COLUMN_3                       0000135E R      03
COLM$K_FAO_XL                    = 0000000D                DDB_NO_ACP                         000013C1 R      03
COLM$K_FAO_XL_NEQ                = 0000008D                DDT$K_LENGTH                     = 00000038
COLM$K_FAO_XW                    = 00000008                DDT$L_ALTSTART                   = 0000001C
COLM$K_LENGTH                    = 00000010                DDT$L_CANCEL                     = 0000000C
CRB$K_LENGTH                     = 00000048                DDT$L_CLONEDUCB                  = 00000024
CRB$L_AUXSTRUC                   = 00000010                DDT$L_FDT                        = 00000008
CRB$L_DUETIME                    = 00000018                DDT$L_MNTVER                     = 00000020
CRB$L_INTD                       = 00000024                DDT$L_REGDUMP                    = 00000010
CRB$L_LINK                       = 00000020                DDT$L_START                      = 00000000
CRB$L_TIMELINK                   = 00000014                DDT$L_UNITINIT                   = 00000018
CRB$L_TOUTROUT                   = 0000001C                DDT$L_UNSOLINT                   = 00000004
CRB$L_WQFL                       = 00000000                DDT$W_DIAGBUF                    = 00000014
CRB$W_REFC                       = 0000000C                DDT$W_ERRORBUF                   = 00000016
CRB_COLUMN_1                       000013F0 R      03      DDT$W_FDTSIZE                    = 00000028
CRB_COLUMN_2                       00001420 R      03      DDT_ADDRESS                        000017B3 R      03
CRB_COLUMN_3                       00001450 R      03      DDT_COLUMN_1                       000016C3 R      03
CRB_DEVCLASS                       00000578 R      02      DDT_COLUMN_2                       00001713 R      03
CRB_TIMEOUT                        00001480 R      03      DDT_COLUMN_3                       00001763 R      03
CROSSED_ASCIC                      00000E3A R      04      DDT_RETURN                         00000B79 R      04
DC$_BUS                          = 00000080                DEFINE_UCB_SYMBOLS                 00001E6A R      03
DC$_CARD                         = 00000041                DEV$M_2P                         = 00000010
DC$_DISK                         = 00000001                DEV$M_DMT                        = 00200000
DC$_JOURNAL                      = 000000A1                DEV$M_MBX                        = 00100000
DC$_LP                           = 00000043                DEV$M_MNT                        = 00080000
DC$_MAILBOX                      = 000000A0                DEV$M_NET                        = 00002000
DC$_MISC                         = 000000C8                DEV$V_2P                         = 00000004
DC$_REALTIME                     = 00000060                DEV$V_ALL                        = 00000017
DC$_SCOM                         = 00000020                DEV$V_AVL                        = 00000012
DC$_TAPE                         = 00000002                DEV$V_CCL                        = 00000001
DC$_TERM                         = 00000042                DEV$V_CDP                        = 00000003
DC$_WORKSTATION                  = 00000046                DEV$V_CLU                        = 00000000
DDB                                00000071 R      02      DEV$V_DET                        = 00000001
DDB$B_ACPCLASS                   = 00000013                DEV$V_DIR                        = 00000003
DDB$B_TYPE                       = 0000000A                DEV$V_DMT                        = 00000015
DDB$C_LENGTH                     = 00000044                DEV$V_DUA                        = 0000000F
DDB$K_CART                       = 00000002                DEV$V_ELG                        = 00000016
DDB$K_LENGTH                     = 00000044                DEV$V_FOD                        = 0000000E
DDB$K_PACK                       = 00000001                DEV$V_FOR                        = 00000018
DDB$K_SLOW                       = 00000003                DEV$V_GEN                        = 00000011
DDB$K_TAPE                       = 00000004                DEV$V_IDV                        = 0000001A
```

```
DEV$V_MBX                  = 00000014                 DT$_DN11                   = 00000001
DEV$V_MNT                  = 00000013                 DT$_DR11C                  = 00000007
DEV$V_MSCP                 = 00000005                 DT$_DR11W                  = 00000004
DEV$V_NET                  = 0000000D                 DT$_DR750                  = 00000003
DEV$V_NNM                  = 00000009                 DT$_DR780                  = 00000002
DEV$V_ODV                  = 0000001B                 DT$_DZ11                   = 00000042
DEV$V_OPR                  = 00000007                 DT$_DZ32                   = 00000043
DEV$V_RCK                  = 0000001E                 DT$_DZ730                  = 00000044
DEV$V_RCT                  = 00000008                 DT$_FT1                    = 00000010
DEV$V_REC                  = 00000000                 DT$_FT2                    = 00000011
DEV$V_RED                  = 00000008                 DT$_FT3                    = 00000012
DEV$V_RND                  = 0000001C                 DT$_FT4                    = 00000013
DEV$V_RTM                  = 0000001D                 DT$_FT5                    = 00000014
DEV$V_RTT                  = 00000002                 DT$_FT6                    = 00000015
DEV$V_SDI                  = 00000004                 DT$_FT7                    = 00000016
DEV$V_SHR                  = 00000010                 DT$_FT8                    = 00000017
DEV$V_SPL                  = 00000006                 DT$_IX_IEX11               = 0000000A
DEV$V_SQD                  = 00000005                 DT$_LAT1                   = 00000002
DEV$V_SRV                  = 00000007                 DT$_LA12                   = 00000024
DEV$V_SSM                  = 00000006                 DT$_LA120                  = 00000021
DEV$V_SWL                  = 00000019                 DT$_LA180                  = 00000003
DEV$V_TRM                  = 00000002                 DT$_LA24                   = 00000025
DEV$V_WCK                  = 0000001F                 DT$_LA34                   = 00000022
DEVICE_CHAR                  00000160 R      03       DT$_LA36                   = 00000020
DEVICE_CHAR_2                00000248 R      03       DT$_LA38                   = 00000023
DEVICE_CLASS                 000002A0 R      03       DT$_LAX                    = 00000020
DISK_TYPE                    00000308 R      03       DT$_LESI                   = 00000005
DISPLAY                      000031B4 R      03       DT$_LP11                   = 00000001
DISPLAY_DDT                  00001275 R      03       DT$_LPA11                  = 00000001
DISPLAY_DEVBYADDR            00000C00 RG     03       DT$_LQP02                  = 00000026
DISPLAY_DEVICE               00000CE2 RG     03       DT$_MBX                    = 00000001
DO_UCB_COLUMNS               00001F25 R      03       DT$_ML11                   = 00000011
DPT                          00000439 R      02       DT$_MX_MUX200              = 00000008
DPT$C_LENGTH               = 00000038                 DT$_NI                     = 0000000D
DPT$L_FLINK                = 00000000                 DT$_NULL                   = 00000003
DPT$T_NAME                 = 00000020                 DT$_NV_X29                 = 00000006
DPT$W_SIZE                 = 00000008                 DT$_NW_X25                 = 00000005
DT$_AIJNL                  = 00000003                 DT$_PCL11R                 = 00000005
DT$_ATJNL                  = 00000004                 DT$_PCL11T                 = 00000006
DT$_BIJNL                  = 00000002                 DT$_RA60                   = 00000016
DT$_CI                     = 0000000C                 DT$_RA80                   = 00000014
DT$_CI750                  = 00000002                 DT$_RA81                   = 00000015
DT$_CI780                  = 00000001                 DT$_RA82                   = 0000001E
DT$_CLJNL                  = 00000005                 DT$_RB02                   = 00000012
DT$_CR11                   = 00000001                 DT$_RB80                   = 00000013
DT$_CRX50                  = 00000021                 DT$_RC25                   = 00000017
DT$_DELUA                  = 00000019                 DT$_RC26                   = 0000001F
DT$_DEQNA                  = 00000016                 DT$_RCF25                  = 00000018
DT$_DEUNA                  = 0000000E                 DT$_RCF26                  = 00000020
DT$_DHU                    = 00000047                 DT$_RD26                   = 0000001D
DT$_DHV                    = 00000046                 DT$_RD51                   = 00000019
DT$_DMC11                  = 00000001                 DT$_RD52                   = 0000001B
DT$_DMF32                  = 0000000A                 DT$_RD53                   = 0000001C
DT$_DMP11                  = 00000009                 DT$_RDRX                   = 00000007
DT$_DMR11                  = 00000002                 DT$_RK06                   = 00000001
DT$_DMV11                  = 00000017                 DT$_RK07                   = 00000002
DT$_DMZ32                  = 00000045                 DT$_RL01                   = 00000009
```

```
DT$_RL02                        = 0000000A            DT$_XV_3271                     = 0000000B
DT$_RM03                        = 00000006            DT$_YN_X25                      = 0000000F
DT$_RM05                        = 0000000F            DT$_YO_X25                      = 00000010
DT$_RM80                        = 0000000D            DT$_YP_ADCCP                    = 00000011
DT$_RP04                        = 00000003            DT$_YQ_3271                     = 00000012
DT$_RP05                        = 00000004            DT$_YR_DDCMP                    = 00000013
DT$_RP06                        = 00000005            DT$_YS_SDLC                     = 00000014
DT$_RP07                        = 00000007            DYN$C_CD_CDDB                   = 00000001
DT$_RP07HT                      = 00000008            DYN$C_CLASSDRV                  = 00000064
DT$_RUJNL                       = 00000001            DYN$C_DDB                       = 00000006
DT$_RX01                        = 00000010            DYN$C_SCS                       = 00000060
DT$_RX02                        = 0000000B            DYN$C_SCS_PDT                   = 00000005
DT$_RX04                        = 0000000C            DYN$C_SCS_SB                    = 00000007
DT$_RX50                        = 0000001A            DYN$C_UCB                       = 00000010
DT$_RZ01                        = 00000017            DYN$C_VCB                       = 00000011
DT$_RZF01                       = 00000018            END_PB                            00001913 R      03
DT$_SB_ISB11                    = 00000007            FAB$L_STV                         ********   X    03
DT$_SHRMBX                      = 00000002            FIND_DPT                          00000EB7 R      03
DT$_TA78                        = 00000006            FLAG_2ND_CDDB                     00000575 R      02
DT$_TA81                        = 00000009            FLAG_M_ACT_PATH                 = 00000002
DT$_TE16                        = 00000001            FLAG_M_FND_UNIT                 = 00000004
DT$_TEK401X                     = 0000000A            FLAG_M_ONE_UNIT                 = 00000001
DT$_TK50                        = 0000000A            FLAG_V_ALT_PATH                 = 00000001
DT$_TQ_BTS                      = 00000004            FLAG_V_FND_UNIT                 = 00000002
DT$_TST1                        = 00000004            FLAG_V_ONE_UNIT                 = 00000000
DT$_TTYUNKN                     = 00000000            FOUND_DPT                         000008D2 R      04
DT$_TU45                        = 00000002            GETMEM                            ********   X    03
DT$_TU58                        = 0000000E            GET_DDB                           00000F05 R      03
DT$_TU77                        = 00000003            GET_UCB                           00001F89 R      03
DT$_TU78                        = 00000005            IDB$B_VECTOR                    = 0000000B
DT$_TU80                        = 00000007            IDB$K_LENGTH                    = 00000038
DT$_TU81                        = 00000008            IDB$L_ADP                       = 00000014
DT$_TU81P                       = 00000006            IDB$L_CSR                       = 00000000
DT$_UDA50                       = 00000003            IDB$L_OWNER                     = 00000004
DT$_UDA50A                      = 00000004            IDB$W_UNITS                     = 0000000C
DT$_UK_KTC32                    = 00000015            IDB_COLUMN_1                      0000162A R      03
DT$_UQPORT                      = 00000003            IDB_COLUMN_2                      0000165A R      03
DT$_VK100                       = 00000002            IDB_COLUMN_3                      0000168A R      03
DT$_VS100                       = 00000001            IDB_VECTOR                        000016AA R      03
DT$_VS125                       = 00000002            IO$S_FCODE                      = 00000006
DT$_VS300                       = 00000003            IO$V_FCODE                      = 00000000
DT$_VT05                        = 00000001            IO$_ACCESS                      = 00000032
DT$_VT100                       = 00000060            IO$_ACPCONTROL                  = 00000038
DT$_VT101                       = 00000061            IO$_AVAILABLE                   = 00000011
DT$_VT102                       = 00000062            IO$_CREATE                      = 00000033
DT$_VT105                       = 00000063            IO$_DEACCESS                    = 00000034
DT$_VT125                       = 00000064            IO$_DELETE                      = 00000035
DT$_VT131                       = 00000065            IO$_DSE                         = 00000015
DT$_VT132                       = 00000066            IO$_ERASETAPE                   = 00000006
DT$_VT173                       = 00000003            IO$_MODIFY                      = 00000036
DT$_VT52                        = 00000040            IO$_NOP                         = 00000000
DT$_VT55                        = 00000041            IO$_PACKACK                     = 00000008
DT$_VT5X                        = 00000040            IO$_READLBLK                    = 00000021
DT$_XI_DR11C                    = 0000000D            IO$_READPBLK                    = 0000000C
DT$_XJ_2780                     = 00000004            IO$_READVBLK                    = 00000031
DT$_XK_3271                     = 00000003            IO$_RECAL                       = 00000003
DT$_XP_PCL11B                   = 00000009            IO$_REWIND                      = 00000024
```

```
IOS_REWINDOFF                  = 00000022             MSCP$V_CF_576                  = 00000000
IO$_SEEK                       = 00000002             MSCP$V_CF_ATTN                 = 00000007
IO$_SENSECHAR                  = 0000001B             MSCP$V_CF_MISC                 = 00000006
IO$_SENSEMODE                  = 00000027             MSCP$V_CF_MLTHS                = 00000002
IO$_SETCHAR                    = 0000001A             MSCP$V_CF_OTHER                = 00000005
IO$_SETMODE                    = 00000023             MSCP$V_CF_REPLC                = 0000000F
IO$_SKIPFILE                   = 00000025             MSCP$V_CF_SHADW                = 00000001
IO$_SKIPRECORD                 = 00000026             MSCP$V_CF_THIS                 = 00000004
IO$_SPACERECORD                = 00000009             MSG$_SUCCESS                     ********   X   03
IO$_UNLOAD                     = 00000001             NEW_PAGE                         ********   X   03
IO$_WRITECHECK                 = 0000000A             NODNAM_2P                        00000060 R     02
IO$_WRITELBLK                  = 00000020             NO_DPT                           00000915 R     04
IO$_WRITEMARK                  = 0000001C             NULL_ASCIC                       00000E21 R     04
IO$_WRITEOF                    = 00000028             OK_ASCIC                         00000E33 R     04
IO$_WRITEPBLK                  = 0000000B             ONE_PATH                         00001065 R     04
IO$_WRITEVBLK                  = 00000030             ORB$L_OWNER                    = 00000000
IO$_WRTTMKR                    = 0000001D             ORB$W_UICGROUP                 = 00000002
IOC$GL_DPTLIST                   ********   X   03    ORB$W_UICMEMBER                = 00000000
IOC$RETURN                       ********   X   03    ORB_OWNER                        00002355 R     03
IO_FUNCTION                      00000A88 R     03    OUTPUT                           ********   X   03
IRP                              000001C5 R     02    PAGE_SIZE                        ********   X   03
```

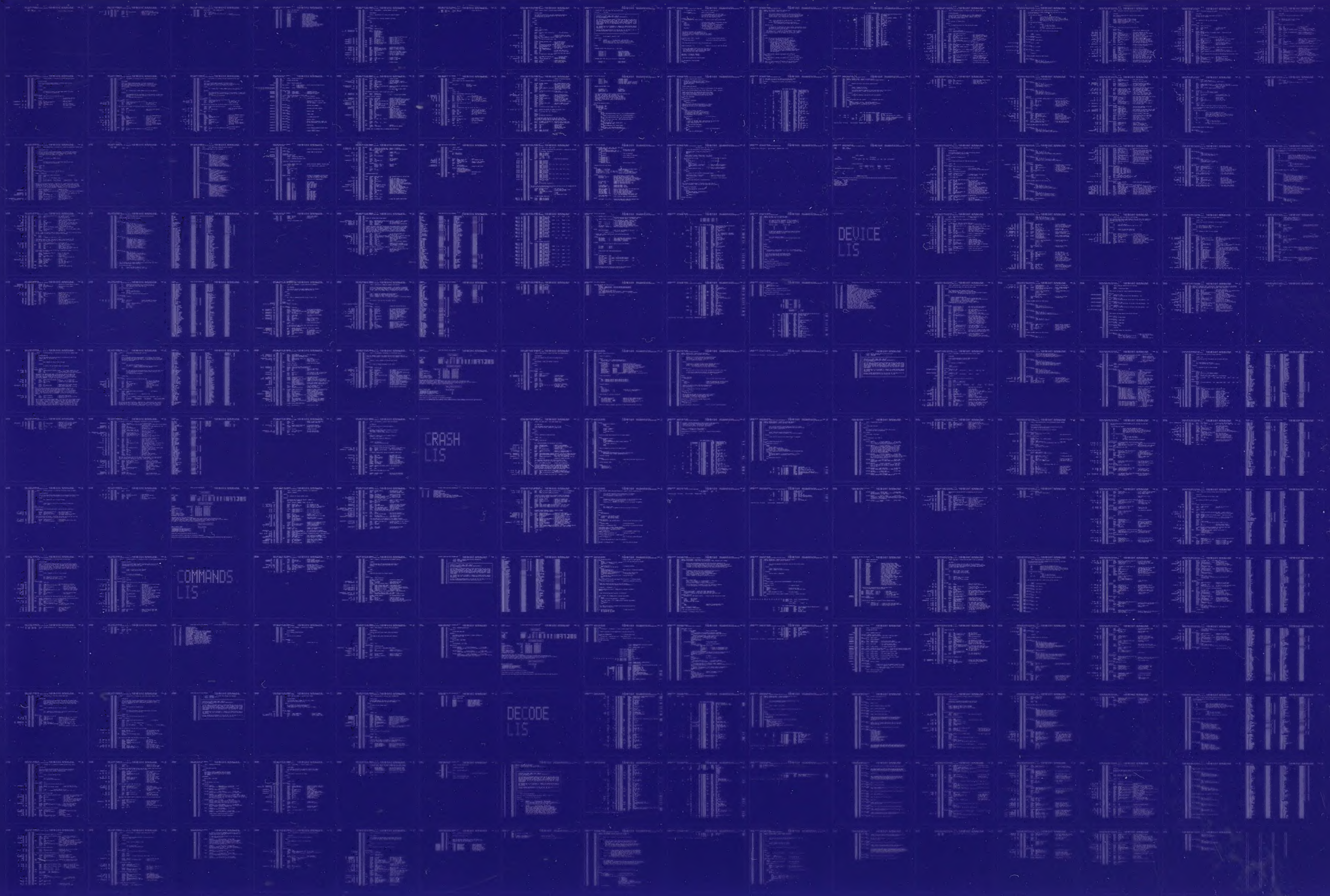
```
PB$V_MAINT              = 00000000                  SB$W_MAXMSG             = 00000022
PB$V_PORT_TYP           = 00000000                  SB$W_TIMEOUT            = 00000058
PB$V_STATE              = 00000001                  SB_6BYTES                 000019F4 R      03
PB$V_TIM                = 00000000                  SB_COLUMN_1               00001914 R      03
PB$W_RETRY              = 00000022                  SB_COLUMN_2               00001984 R      03
PB$W_STATE              = 00000012                  SB_FAO_6BYTES             00000DEE R      04
PB$W_STS                = 00000044                  SB_FAO_ASCIC              00000E00 R      04
PB_CABLES                 00001BD6 R      03        SB_LWCHAR                 00001A14 R      03
PB_COLUMN_1               00001A38 R      03        SCH$GL_PCBVEC             ********   X    03
PB_COLUMN_2               00001A88 R      03        SCOM_TYPE                 00000480 R      03
PB_DUALPATH               00001BBD R      03        SCS$GA_LOCALSB            ********   X    03
PB_LCLSTATE               00001C51 R      03        SCS$GQ_CONFIG             ********   X    03
PB_LOOP                   00001854 R      03        SECOND                    000031A7 R      03
PB_RMTSTATE               00001B38 R      03        SETUP_PRIMARY             00001DC4 R      03
PB_RPORT_TYP              00001BBA R      03        SET_HEADING               ********   X    03
PB_RPORT_TYPE             00000058 R      03        SHOW_ACPQ                 000025B3 R      03
PB_RSTATE                 00000038 R      03        SHOW_CDDB                 0000313C R      03
PB_STATE                  00000010 R      03        SHOW_CONTROLLER           00000FE1 R      03
PB_STATUS                 00000000 R      03        SHOW_DDBS                 00000DFD R      03
PCB$T_LNAME             = 00000070                  SHOW_IOQ                  000024C9 R      03
PDVNM_B_NODESZ            00000022                  SHOW_SYSTEM_BLOCK         000017D8 RG     03
PDVNM_K_LENGTH          = 00000024                  SHOW_UCB                  00001C7B R      03
PDVNM_T_DDC               00000010                  SHOW_VCB                  00002AB1 R      03
PDVNM_T_NODE              00000000                  SIZ...                  = 00000001
PDVNM_W_UNIT              00000020                  SKIP_LINES                ********   X    03
PRINT                     ********   X    03        SKIP_SB                   0000103A R      03
PRINT_CDRP                00002823 R      03        SKIP_SECOND_CRB           000011E8 R      03
PRINT_COLUMNS             ********   X    03        SS$_NOSUCHDEV             ********   X    03
PRINT_COLUMN_VALUE        ********   X    03        SYS$FAO                   ********   X    03
PRINT_IRP                 000029D0 R      03        SYS$FAOL                  ********  GX    03
PROCESS_2P_DDB            00001D7C R      03        SYS$PUT                   ********  GX    03
QUEUE_NOTEMPTY            00000577 R      02        TAPE_TYPE                 00000428 R      03
QUEUE_TITLE               00002962 R      03        TERM_TYPE                 00000558 R      03
RAB$L_RBF                 ********   X    03        THIS_PRIMARY              0000109B R      04
RAB$W_RSZ                 ********   X    03        THIS_SECONDARY            000010DD R      04
REALTIME_TYPE             000006D0 R      03        TPA$C_NUMBER            = 0000001C
REQUEST_STATUS            00000A10 R      03        TPA$L_TOKENCNT          = 00000010
RETRY_CNT                 00003419 R      03        TRANSLATE_ADDRESS         ********   X    03
RSTRT_CDRP                00003405 R      03        TRANSLATE_BITS            ********   X    03
SB                        00000000 R      02        TRYMEM                    ********   X    03
SB$B_ENBMSK             = 0000005A                  UCB                       000000F9 R      02
SB$B_HWVERS             = 00000038                  UCB$B_DEVCLASS          = 00000040
SB$B_SYSTEMID           = 00000018                  UCB$B_DEVTYPE           = 00000041
SB$B_TYPE               = 0000000A                  UCB$B_DIPL              = 0000005E
SB$C_LENGTH             = 00000060                  UCB$B_ERTCNT            = 00000080
SB$K_LENGTH             = 00000060                  UCB$B_ERTMAX            = 00000081
SB$L_DDB                = 00000054                  UCB$B_FIPL              = 0000000B
SB$L_FLINK              = 00000000                  UCB$B_ONLCNT            = 000000AE
SB$L_PBFL               = 0000000C                  UCB$B_TYPE              = 0000000A
SB$Q_SWINCARN           = 0000002C                  UCB$K_LCL_DISK_LENGTH   = 000000CC
SB$Q_SWINCARN2          = 00000030                  UCB$L_2P_CDDB           = 000000C0
SB$S_NODENAME           = 00000010                  UCB$L_AMB               = 00000060
SB$T_HWTYPE             = 00000034                  UCB$L_CDDB              = 000000BC
SB$T_NODENAME           = 00000044                  UCB$L_CPID              = 00000020
SB$T_SWTYPE             = 00000024                  UCB$L_CRB               = 00000024
SB$T_SWVERS             = 00000028                  UCB$L_DDB               = 00000028
SB$W_MAXDG              = 00000020                  UCB$L_DDT               = 00000088
```

```
UCB$L_DEVCHAR                   = 00000038          UCB_2PDDB                       000024A4 R      03
UCB$L_DEVCHAR2                  = 0000003C          UCB_ACT_NOP                     000022D5 R      03
UCB$L_DEVDEPEND                 = 00000044          UCB_ACT_NOP_A                   000023BD R      03
UCB$L_DEVDEPND2                 = 00000048          UCB_ACT_NOP_B                   000024C6 R      03
UCB$L_DP_ALTUCB                 = 000000A8          UCB_ACT_UB                      0000227A R      03
UCB$L_DP_DDB                    = 000000A0          UCB_ACT_XL                      00002330 R      03
UCB$L_DP_LINK                   = 000000A4          UCB_ACT_XL_NEQ                  000022B1 R      03
UCB$L_DUETIM                    = 0000006C          UCB_ACT_XW                      00002467 R      03
UCB$L_FPC                       = 0000000C          UCB_ALLOCLASS                   00002271 R      03
UCB$L_FR3                       = 00000010          UCB_ALTUCB                      00002283 R      03
UCB$L_FR4                       = 00000014          UCB_BSY                         000022A9 R      03
UCB$L_IOQFL                     = 0000004C          UCB_CDDB                        000023C0 R      03
UCB$L_IRP                       = 00000058          UCB_CLSTYP                      000022BB R      03
UCB$L_JNL_MCSID                 = 00000084          UCB_COLUMN_1                    00001FC1 R      03
UCB$L_LINK                      = 00000030          UCB_COLUMN_2                    00002071 R      03
UCB$L_LOCKID                    = 00000020          UCB_COLUMN_3                    00002151 R      03
UCB$L_LOGADR                    = 00000074          UCB_CPID                        000022C5 R      03
UCB$L_OPCNT                     = 00000070          UCB_DDB                         0000057C R      02
UCB$L_ORB                       = 0000001C          UCB_DUETIM                      000022D8 R      03
UCB$L_PDT                       = 00000084          UCB_IPLS                        000022E3 R      03
UCB$L_PID                       = 0000002C          UCB_LNM                         00002301 R      03
UCB$L_STS                       = 00000064          UCB_LOCKID                      00002327 R      03
UCB$L_SVAPTE                    = 00000078          UCB_MCSID                       00002339 R      03
UCB$L_SVPN                      = 00000074          UCB_ONLCNT                      00002347 R      03
UCB$L_TL_PHYUCB                 = 000000A0          UCB_PDT                         00002388 R      03
UCB$L_VCB                       = 00000034          UCB_RETRY                       00002408 R      03
UCB$V_BSY                       = 00000008          UCB_RETRY_FAO                   0000118C R      04
UCB$V_CANCEL                    = 00000003          UCB_RET_2XBYTES                 000022EB R      03
UCB$V_DEADMO                    = 0000000A          UCB_RWAITCNT                    00002449 R      03
UCB$V_DELETEUCB                 = 00000010          UCB_SIZE                      = 000000CC
UCB$V_ERLOGIP                   = 00000002          UCB_SVPN                        00002470 R      03
UCB$V_INT                       = 00000001          UCB_TEST_RETRY_FAO              0000119C R      04
UCB$V_INTTYPE                   = 00000007          UCB_TWO_BYTES                   0000117B R      04
UCB$V_LCL_VALID                 = 00000011          UCB_UIC_CSTR1                   00001168 R      04
UCB$V_MNTVERIP                  = 0000000E          UCB_VCB                         0000247E R      03
UCB$V_MNTVERPND                 = 00000013          UNIT_STATUS                     000000B8 R      03
UCB$V_MOUNTING                  = 00000009          UNKNOWN                         00001160 R      04
UCB$V_ONLINE                    = 00000004          VCB                             00000331 R      02
UCB$V_POWER                     = 00000005          VCB$B_CUR_RVN                 = 0000002F
UCB$V_SUPMVMSG                  = 00000012          VCB$B_JNL_MODE                = 00000044
UCB$V_TEMPLATE                  = 0000000D          VCB$B_RESFILES                = 0000004F
UCB$V_TIM                       = 00000000          VCB$B_STATUS                  = 0000000B
UCB$V_TIMOUT                    = 00000006          VCB$B_STATUS2                 = 00000053
UCB$V_UNLOAD                    = 0000000C          VCB$B_TM                      = 0000002E
UCB$V_VALID                     = 0000000B          VCB$B_TYPE                    = 0000000A
UCB$V_WRONGVOL                  = 0000000F          VCB$B_WINDOW                  = 00000048
UCB$W_BCNT                      = 0000007E          VCB$C_LENGTH                  = 000000EC
UCB$W_BOFF                      = 0000007C          VCB$L_AQB                     = 00000010
UCB$W_DEVBUFSIZ                 = 00000042          VCB$L_BLOCKFL                 = 00000000
UCB$W_DEVSTS                    = 00000068          VCB$L_BLOCKID                 = 0000008C
UCB$W_ERRCNT                    = 00000082          VCB$L_CACHE                   = 00000058
UCB$W_REFC                      = 0000005C          VCB$L_FCBFL                   = 00000000
UCB$W_RWAITCNT                  = 00000056          VCB$L_FREE                    = 00000040
UCB$W_SIZE                      = 00000008          VCB$L_JNL_CHAR                = 00000024
UCB$W_STS                       = 00000064          VCB$L_JNL_JFTA                = 00000028
UCB$W_UNIT                      = 00000054          VCB$L_JNL_JMT                 = 00000034
UCB_2PCDDB                        000023E2 R      03  VCB$L_JNL_MASK                = 00000048
```

```
VCB$L_MAXFILES                  = 00000044                 VCB_DISK_COL_2                    00002EBC R       03
VCB$L_MVL                       = 00000034                 VCB_DISK_COL_3                    00002F2C R       03
VCB$L_QUOCACHE                  = 0000005C                 VCB_DISK_STATUS                   000007D8 R       03
VCB$L_QUOTAFCB                  = 00000054                 VCB_DISK_STATUS2                  00000820 R       03
VCB$L_RVT                       = 00000020                 VCB_FOREIGN                       00002D1D R       03
VCB$L_ST_RECORD                 = 00000030                 VCB_JNL_COL_1                     000030AC R       03
VCB$L_VOLCKID                   = 0000007C                 VCB_JNL_COL_2                     000030DC R       03
VCB$L_VPFL                      = 0000003C                 VCB_JNL_COL_3                     0000310C R       03
VCB$T_VOLCKNAM                  = 00000080                 VCB_JOURNAL                       00002DE9 R       03
VCB$T_VOLNAME                   = 00000014                 VCB_JOURNAL_CHAR                  00000908 R       03
VCB$V_BLANK                     = 0000000A                 VCB_NET                           00002DC8 R       03
VCB$V_CANCELIO                  = 00000005                 VCB_NET_COL_1                     0000304C R       03
VCB$V_EBCDIC                    = 00000005                 VCB_NET_COL_2                     0000306C R       03
VCB$V_ENUSEREOT                 = 00000009                 VCB_NET_COL_3                     0000308C R       03
VCB$V_ERASE                     = 00000003                 VCB_SHOW_ACPQ                     00002E46 R       03
VCB$V_EXTFID                    = 00000005                 VCB_TAPE                          00002D1D R       03
VCB$V_GROUP                     = 00000006                 VCB_TAPE_COL_1                    00002F9C R       03
VCB$V_HOMBLKBAD                 = 00000002                 VCB_TAPE_COL_2                    00002FDC R       03
VCB$V_IDXHDRBAD                 = 00000003                 VCB_TAPE_COL_3                    0000300C R       03
VCB$V_INIT                      = 0000000B                 VCB_TAPE_MODE                     00000898 R       03
VCB$V_INTCHG                    = 00000004                 VCB_TAPE_STATUS                   00000850 R       03
VCB$V_JNL_DISK                  = 00000000                 VEC$B_DATAPATH                  = 00000013
VCB$V_JNL_TAPE                  = 00000001                 VEC$B_NUMREG                    = 00000012
VCB$V_JNL_TMPFI                 = 00000002                 VEC$L_ADP                       = 00000014
VCB$V_LOGICEOVS                 = 00000001                 VEC$L_IDB                       = 00000008
VCB$V_MOUNTVER                  = 00000002                 VEC$L_INITIAL                   = 0000000C
VCB$V_MUSTCLOSE                 = 00000006                 VEC$L_INTSER                    = 00000004
VCB$V_NOALLOC                   = 00000004                 VEC$L_START                     = 0000001C
VCB$V_NOAUTO                    = 0000000C                 VEC$L_UNITDISC                  = 00000020
VCB$V_NOCACHE                   = 00000001                 VEC$L_UNITINIT                  = 00000018
VCB$V_NOHIGHWATER               = 00000004                 VEC$Q_DISPATCH                  = 00000000
VCB$V_NOVOL2                    = 00000006                 VEC$S_DATAPATH                  = 00000005
VCB$V_NOWRITE                   = 00000007                 VEC$S_MAPREG                    = 0000000F
VCB$V_OVRACC                    = 00000001                 VEC$V_DATAPATH                  = 00000000
VCB$V_OVREXP                    = 00000000                 VEC$V_LWAE                      = 00000005
VCB$V_OVRLBL                    = 00000002                 VEC$V_MAPLOCK                   = 0000000F
VCB$V_OVRSETID                  = 00000003                 VEC$V_MAPREG                    = 00000000
VCB$V_OVRVOLO                   = 0000000D                 VEC$V_PATHLOCK                  = 00000007
VCB$V_PARTFILE                  = 00000000                 VEC$W_MAPREG                    = 00000010
VCB$V_STARFILE                  = 00000008                 VEC_COLUMN_1                      0000149E R       03
VCB$V_SYSTEM                    = 00000007                 VEC_COLUMN_2                      000014DE R       03
VCB$V_WAIMOUVOL                 = 00000002                 VEC_COLUMN_3                      0000151E R       03
VCB$V_WAIREWIND                 = 00000003                 VEC_DATAPATH                      0000155E R       03
VCB$V_WAIUSRLBL                 = 00000004                 VEC_FAO_DATAPATH                  00000B47 R       04
VCB$V_WRITETHRU                 = 00000000                 VEC_FAO_MAPREG                    00000B58 R       04
VCB$V_WRITE_IF                  = 00000000                 VEC_LOCKED                        00000B71 R       04
VCB$V_WRITE_SM                  = 00000001                 VEC_LWAE                          00000B6B R       04
VCB$W_CLUSTER                   = 0000003C                 VEC_MAPREG                        000015DC R       03
VCB$W_EXTEND                    = 0000003E                 VEC_TEST_UBA                      000015BE R       03
VCB$W_JNL_COP                   = 00000045                 VIRTUAL_TERMINAL                  0000111F R       04
VCB$W_MCOUNT                    = 0000004C                 WORKSTATION_TYPE                  000006B0 R       03
VCB$W_MODE                      = 0000002C
VCB$W_RECORDSZ                  = 00000050
VCB$W_RVN                       = 0000000E
VCB$W_TRANS                     = 0000000C
VCB_DISK                          00002C8B R       03
VCB_DISK_COL_1                    00002E4C R       03
```

```
                                             +----------------+
                                             ! Psect synopsis !
                                             +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000024  (    36.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
SDADATA                         00000580  (  1408.)  02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR NOEXE   RD    WRT NOVEC BYTE
DEVICE                          0000342A  (13354.)  03 (  3.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC LONG
LITERALS                        00001B70  ( 7024.)  04 (  4.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD  NOWRT NOVEC BYTE

                                          +------------------------+
                                          ! Performance indicators !
                                          +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     29    00:00:00.04    00:00:00.99
Command processing                108    00:00:00.41    00:00:03.28
Pass 1                           1290    00:00:43.46    00:02:42.73
Symbol table sort                   0    00:00:03.73    00:00:14.02
Pass 2                            919    00:00:10.33    00:00:34.74
Symbol table output                 1    00:00:00.44    00:00:01.57
Psect synopsis output               0    00:00:00.02    00:00:00.01
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals             2349    00:00:58.45    00:03:37.38

The working set limit was 3000 pages.
381141 bytes (745 pages) of virtual memory were used to buffer the intermediate code.
There were 190 pages of symbol table space allocated to hold 3185 non-local and 393 local symbols.
2914 source lines were read in Pass 1, producing 108 object records in Pass 2.
69 pages of virtual memory were used to define 64 macros.

                                         +--------------------------+
                                         ! Macro library statistics !
                                         +--------------------------+

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[SDA.OBJ]SDALIB.MLB;1                    20
_$255$DUA28:[SYS.OBJ]LIB.MLB;1                       21
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    14
TOTALS (all libraries)                               55

3409 GETS were required to define 55 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:DEVICE/OBJ=OBJ$:DEVICE MSRC$:DEVICE/UPDATE=(ENH$:DEVICE)+EXECML$/LIB+LIB$:SDALIB/LIB
```

0351 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY



DEVICE
LIS

CRASH
LIS

COMMANDS
LIS

DECODE
LIS

0352 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY